INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN

Plan de estudios en extinción

CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN

CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

Plan de estudios NUEVO

CÓDIGO CARRERA: 53=SISTEMAS y 54=GESTIÓN

CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

NACIONAL, 1ª SEMANA

Departamento de Lenguajes y Sistemas Informáticos

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

FECHA: 26 de mayo de 2004 Hora: 11:30 Duración: 2 horas MATERIAL: NINGUNO

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) la nota mínima es 1 punto; en la segunda parte (ejercicio de teoría aplicada que se valora con 5 puntos) la nota mínima que debe obtener es de 2 puntos.

¡ATENCIÓN! PONGA SUS DATOS EN LA HOJA DE LECTURA ÓPTICA QUE DEBERÁ ENTREGAR JUNTO CON EL RESTO DE LAS RESPUESTAS.

Conteste a las preguntas teóricas, en cualquier orden, en hojas diferentes a las que utilice para la contestación de la segunda parte. En cada parte, la cantidad MÁXIMA de papel (de examen, timbrado) que puede emplear ESTÁ LIMITADA al equivalente a DOS (2) HOJAS de tamaño A4 (210 x 297 mm)

PRIMERA PARTE. PREGUNTAS TEÓRICAS (2'5 PUNTOS CADA UNA)

1. Defina y distinga la 'validación' y la 'verificación'. ¿En qué fase del ciclo de vida de cascada se realiza cada una?

   Páginas 15 y 16 del libro.

2. ¿Qué tres objetivos fundamentales o cualidades mínimas es deseable alcanzar al hacer la descomposición modular de un sistema? Explique cada uno de ellos y, en cada caso, cómo se pueden medir o qué factores intervienen.

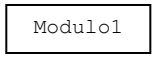
   Reflexión sobre el epígrafe 4.1, páginas 150-160 del libro.

SEGUNDA PARTE. PREGUNTA DE TEORÍA APLICADA (MÁXIMO 5 PUNTOS)

3. Ejercicio de diseño

Dentro de un sistema informático se emplea el módulo `Modulo1`. Para su desarrollo se plantean dos diseños alternativos:

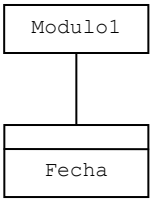
Diseño 1



Código aproximado en Modula-2

```
MODULE Modulo1;
FROM InOut
   IMPORT WriteCard, WriteString;
...
VAR
  fecha1, fecha2:
    ARRAY [1..6] OF TipoDigito;
  ...
BEGIN
  ...
  (* fecha1 := 30-04-1974 *)
  fecha1[1] := 3;
  fecha1[2] := 0;
  fecha1[3] := 0;
  fecha1[4] := 4;
  fecha1[5] := 7;
  fecha1[6] := 4;
  (* Imprimir anno *)
  WriteCard(fecha1[5], 1);
  WriteCard(fecha1[6], 1);
  ...
  (* fecha2 := 10-01-1998 *)
  fecha2[1] := 1;
  fecha2[2] := 0;
  fecha2[3] := 0;
  fecha2[4] := 1;
  fecha2[5] := 9;
  fecha2[6] := 8;
  ...
END Modulo1.
```

Diseño 2



Código aproximado en Modula-2

```
MODULE Modulo1;
IMPORT Fecha;
...
VAR
  fecha1, fecha2: Fecha.Tipo;
  ...
BEGIN
  ...
  Fecha.Crear(fecha1, 30, 4, 1974);
  Fecha.ImprimirAnno(fecha1);
  ...
  Fecha.Crear(fecha2, 10, 1, 1998);
  ...
END Modulo1.
```

```
DEFINITION MODULE Fecha;
TYPE Tipo;
PROCEDURE Crear(VAR fecha: Tipo);
PROCEDURE ImprimirAnno(fecha: Tipo);
...
END Fecha.
```

Compare los dos diseños analizando cómo aplican los conceptos de Abstracción, Modularidad y Ocultación.

Inicialmente, cuando se planteó el sistema informático, se consideró que para el almacenamiento de los años bastaba con dos dígitos. Sin embargo, con la llegada el nuevo milenio se descubrió que eran necesarios cuatro dígitos. Razone como afectaría este cambio a cada uno de los diseños.

SOLUCIÓN

1. Abstracción

- El primer diseño no utiliza abstracciones de ningún tipo.
- El diseño 2 utiliza el Tipo Abstracto de Datos[1] `Fecha`.

Como resultado, se puede observar que el código asociado al diseño 1 será muy redundante (líneas 12-17 ≈ líneas 23-28). La redundancia induce a errores e inconsistencias.

2. Modularidad

El diseño 1 es monolítico, mientras que el diseño 2 es modular. Por ello, el diseño 2 dispone de las siguientes ventajas sobre el diseño 1:

- Permite dividir la implementación entre varias personas (un programador puede codificar `Modulo1` y otro, `Fecha`)
- La implementación asociada al diseño 2 es clara y concisa.
- Los costes asociados al desarrollo, la depuración, la documentación y el mantenimiento del diseño 2 son menores que los del diseño 1.
- El diseño 2 permite reutilizar el concepto de fecha en otros proyectos.

3. Ocultación

En el diseño 1 las "interioridades de las fechas están al descubierto", es decir, no existe ocultación del concepto de fecha. Este hecho, conlleva dos grandes problemas:

- Si se produce un error en el uso de una fecha, su detección será ardua, ya que a este concepto se accede directamente desde muchos puntos del programa.
- La modificación de la representación del concepto fecha se propagará a muchas partes del código del `Modulo1`. Si tal y como se propone en el enunciado, se decide extender la representación de los años de 2 dígitos a 4, será necesario modificar las líneas 7, 12-28.

En el diseño 2, el concepto de fecha se encapsula y oculta a través de un Tipo Abstracto de Datos. Las consecuencias benignas de esta estrategia son:

- Si se produce un error asociado a una fecha, la búsqueda se limitará al módulo de implementación de `Fecha`.
- Gracias a la ocultación del concepto fecha, la ampliación del número de dígitos de los años implica cambios exclusivamente en el código del módulo de implementación de `Fecha`.

[1] Concretamente y según la nomenclatura estudiada en la asignatura Programación 1, el *Tipo Opaco de Datos* `Fecha`.

INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN

Plan de estudios en extinción
CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

Plan de estudios NUEVO
CÓDIGO CARRERA: 53=SISTEMAS y 54=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

NACIONAL 2ª SEMANA

Departamento de Lenguajes y Sistemas Informáticos

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)
FECHA: 9 de junio de 2004 Hora: 11:30 Duración: 2 horas MATERIAL: NINGUNO

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) la nota mínima es 1 punto; en la segunda parte (ejercicio de teoría aplicada que se valora con 5 puntos) la nota mínima que debe obtener es de 2 puntos.

¡ATENCIÓN! PONGA SUS DATOS EN LA HOJA DE LECTURA ÓPTICA QUE DEBERÁ ENTREGAR JUNTO CON EL RESTO DE LAS RESPUESTAS.

Conteste a las preguntas teóricas, en cualquier orden, en hojas diferentes a las que utilice para la contestación de la segunda parte. En cada parte, la cantidad MÁXIMA de papel (de examen, timbrado) que puede emplear ESTÁ LIMITADA al equivalente a DOS (2) HOJAS de tamaño A4 (210 x 297 mm)

PRIMERA PARTE. PREGUNTAS TEÓRICAS (2'5 PUNTOS CADA UNA)

1. Supóngase que una organización decide afrontar, por primera vez, un proyecto software en el que no tiene experiencia y que se sitúa en un ámbito desconocido para ella. Desde el punto de vista del ciclo de vida, indique qué alternativas tiene esta organización para culminar el proyecto con éxito y explique cómo pueden, dichas alternativas, mitigar los problemas potenciales con los que la organización se puede encontrar durante el desarrollo.

   Reflexión sobre 'Modelos del proceso de desarrollo (Ciclos de Vida)'. En concreto, puntos 1.5 y 1.6 del libro.

2. Defina qué es un TAD (Tipo Abstracto de Datos) y qué repercusiones tiene su uso en el diseño de software.

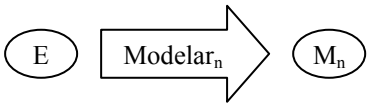
   Tradicionalmente, en la programación imperativa se optaba por separar los programas en dos partes: la de proceso y la de datos (ejemplos de lenguajes imperativos son FORTRAN, COBOL, PASCAL, BASIC…). La parte de proceso accedía y operaba directamente sobre los datos.
   Esta separación produce una independencia funcional muy baja. Un ejemplo que ilustra esto es el "efecto 2000", donde la simple adición de dígitos al formato de las fechas supuso realizar muchas modificaciones en la parte de proceso.
   Para solventar estos problemas surgió el concepto de Tipo Abstracto de Datos (TAD), que agrupa en una sola entidad la representación de los datos y la parte de proceso que los manipula. Los TADs ocultan la representación de los datos, que sólo es accesible desde las operaciones. De esta forma, un cambio en la representación de los datos de un TAD no se propaga a todo el programa, sino solamente a las operaciones del TAD.
   Tradicionalmente, las estructuras de datos se venían definiendo por su representación. Sin embargo, el paso clave hacia la abstracción de los datos es invertir este punto de vista: olvidar por el momento la representación y considerar que las operaciones en sí mismas definen la estructura de datos.

NOTA: un error grave cometido por muchos alumnos consiste en diseñar una operación fuera del TAD que le corresponde. Por ejemplo, sería erróneo el utilizar un TAD llamado "Rotaciones" y una abstracción funcional, separada del TAD, llamada "ProducirRotaciones".

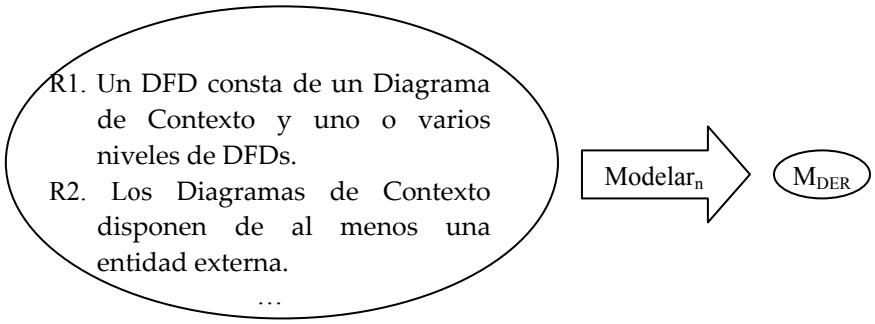
SEGUNDA PARTE. PREGUNTA DE TEORÍA APLICADA (MÁXIMO 5 PUNTOS)

3. **Metamodelar DFD con DER**

Modelar una especificación *E* con una notación *n*, significa obtener un modelo M_n que represente adecuadamente a *E*.

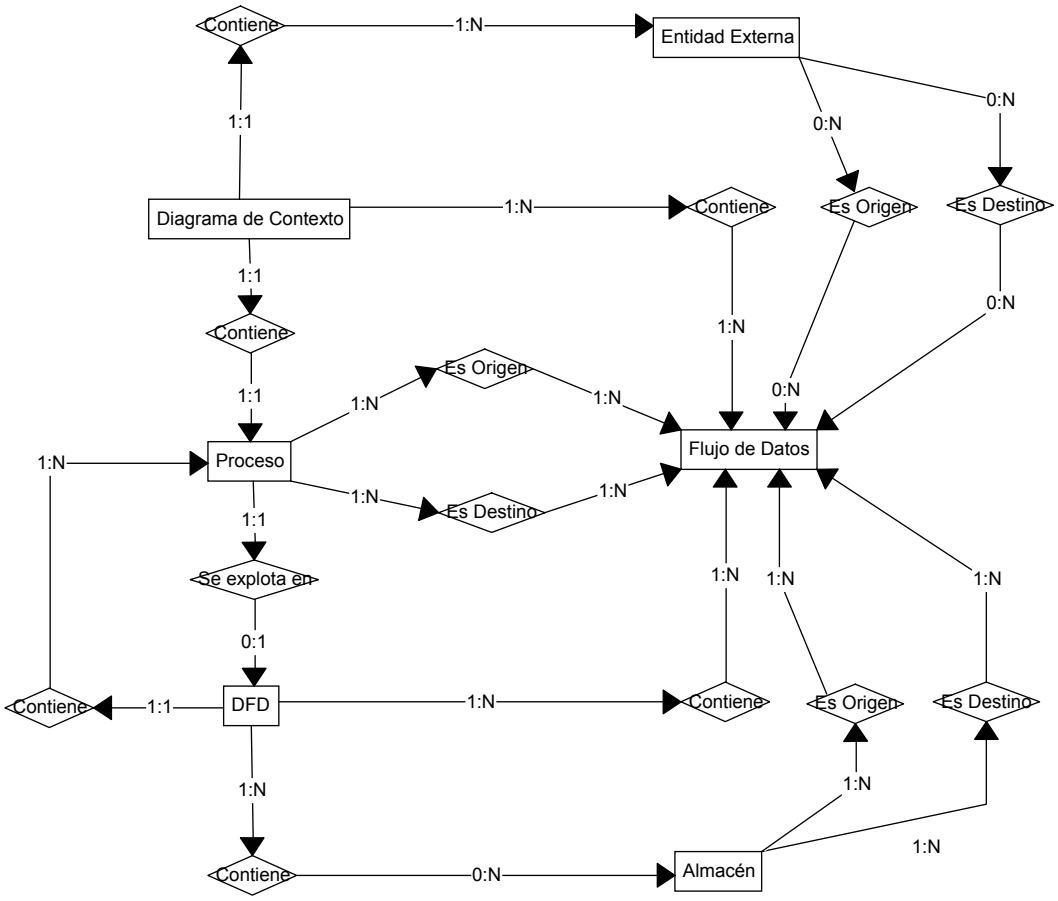


En el caso del metamodelado, la especificación que se utiliza como entrada es también la correspondiente a una notación de modelado.
Metamodele con un DER (Diagrama Entidad Relación) la notación DFD (Diagrama de Flujo de Datos), es decir, para la siguiente figura, desarrolle M_{DER}.



AYUDA: Se sugiere que se haga lo siguiente:
1. Escriba una lista con las especificaciones funcionales que reflejen cómo se construye un DFD.
2. Con la lista anterior, construya el modelo utilizando, para ello, la notación Entidad - Relación.

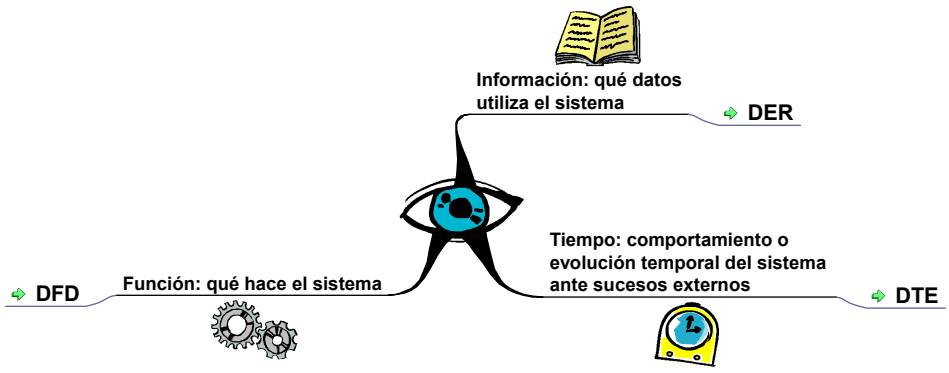
SOLUCIÓN



Observación

Durante la corrección, hemos detectado que algunos alumnos han interpretado incorrectamente el enunciado y han tratado de buscar una equivalencia entre los DFDs y los DERs. Por ejemplo, los flujos de datos[DFD] se corresponden con relaciones[DER], los almacenes de datos[DFD] y las entidades externas[DFD] con entidades[DER]…

Esto supone un grave error, ya que los DFDs, DERs y DTEs representan distintos puntos de vista (prácticamente ortogonales) de un sistema software, y por tanto, es absurdo buscar su equivalencia.



INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN

Plan de estudios en extinción
CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

Plan de estudios NUEVO
CÓDIGO CARRERA: 53=SISTEMAS y 54=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

ORIGINAL EXTRANJERO

UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

Departamento de Lenguajes y Sistemas Informáticos

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)
FECHA: 9 de junio de 2004 Hora: 11:30 Duración: 2 horas MATERIAL: NINGUNO

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) la nota mínima es 1 punto; en la segunda parte (ejercicio de teoría aplicada que se valora con 5 puntos) la nota mínima que debe obtener es de 2 puntos.

¡ATENCIÓN! PONGA SUS DATOS EN LA HOJA DE LECTURA ÓPTICA QUE DEBERÁ ENTREGAR JUNTO CON EL RESTO DE LAS RESPUESTAS.

Conteste a las preguntas teóricas, en cualquier orden, en hojas diferentes a las que utilice para la contestación de la segunda parte. En cada parte, la cantidad MÁXIMA de papel (de examen, timbrado) que puede emplear ESTÁ LIMITADA al equivalente a DOS (2) HOJAS de tamaño A4 (210 x 297 mm)

PRIMERA PARTE. PREGUNTAS TEÓRICAS (2'5 PUNTOS CADA UNA)

1. Defina el concepto de 'configuración software'. Explique qué es y cómo se utiliza la 'línea base' para gestionar los cambios en la gestión de configuración del software.

   *Contenido del epígrafe **1.9.5 Gestión de configuración**, páginas 30 a 32 del libro.*

2. Defina qué es un TAD (Tipo Abstracto de Datos) y qué repercusiones tiene su uso en el diseño de software.

   Tradicionalmente, en la programación imperativa se optaba por separar los programas en dos partes: la de proceso y la de datos (ejemplos de lenguajes imperativos son FORTRAN, COBOL, PASCAL, BASIC…). La parte de proceso accedía y operaba directamente sobre los datos.
   Esta separación produce una independencia funcional muy baja. Un ejemplo que ilustra esto es el "efecto 2000", donde la simple adición de dígitos al formato de las fechas supuso realizar muchas modificaciones en la parte de proceso.
   Para solventar estos problemas surgió el concepto de Tipo Abstracto de Datos (TAD), que agrupa en una sola entidad la representación de los datos y la parte de proceso que los manipula. Los TADs ocultan la representación de los datos, que sólo es accesible desde las operaciones. De esta forma, un cambio en la representación de los datos de un TAD no se propaga a todo el programa, sino solamente a las operaciones del TAD.
   Tradicionalmente, las estructuras de datos se venían definiendo por su representación. Sin embargo, el paso clave hacia la abstracción de los datos es invertir este punto de vista: olvidar por el momento la representación y considerar que las operaciones en sí mismas definen la estructura de datos.

   NOTA: un error grave cometido por muchos alumnos consiste en diseñar una operación fuera del TAD que le corresponde. Por ejemplo, sería erróneo el utilizar un TAD llamado "Rotaciones" y una abstracción funcional, separada del TAD, llamada "ProducirRotaciones".

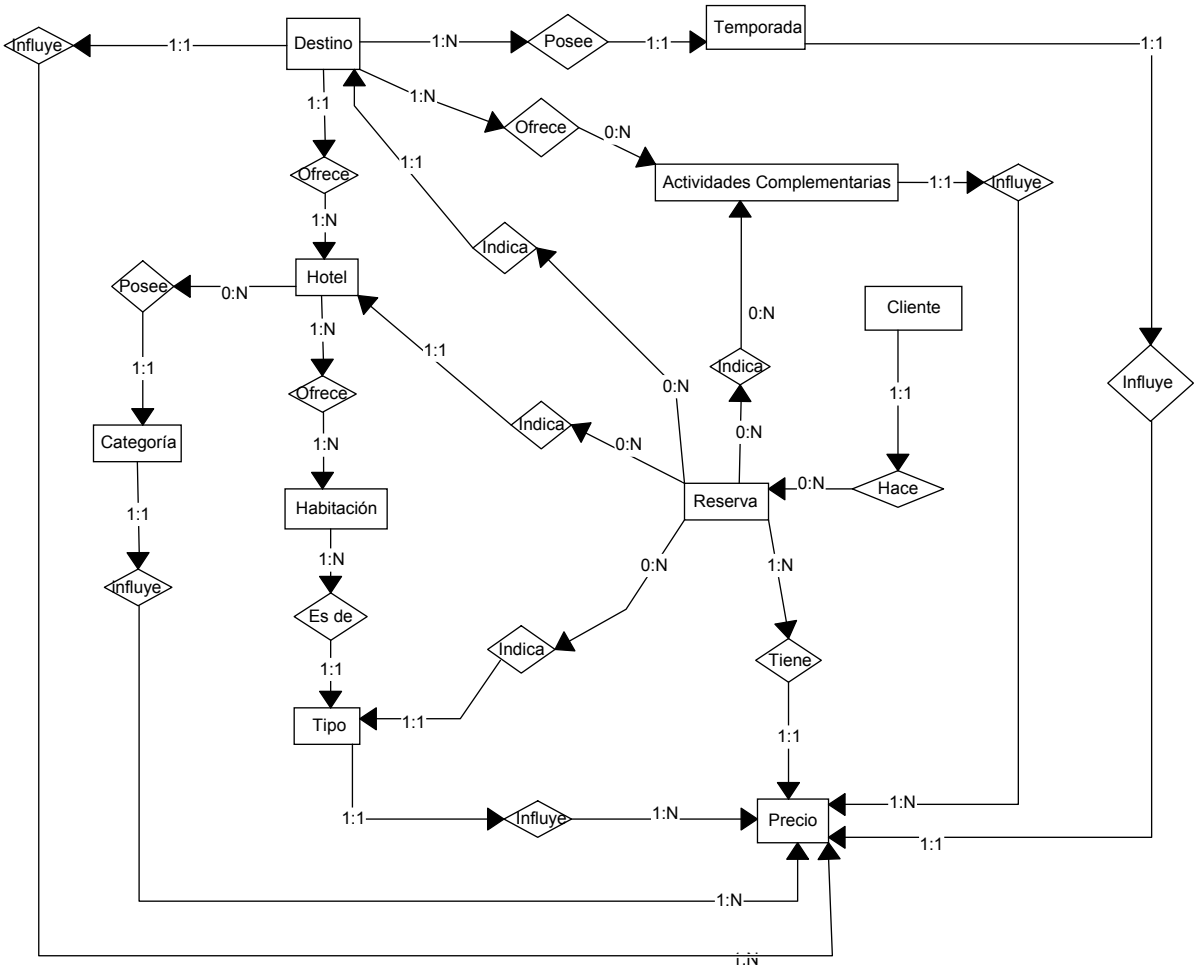
SEGUNDA PARTE. PREGUNTA DE TEORÍA APLICADA (MÁXIMO 5 PUNTOS)

3. Un cliente nos propone desarrollar una aplicación para gestionar algunos aspectos de una agencia de viajes y nos ofrece esta descripción:

   "La agencia de viajes ofrece a sus clientes destinos tanto nacionales como en el extranjero. Para cada destino, la agencia ofrece alojamiento en varios hoteles de distinta categoría. Así mismo, se ofrece, para cada destino, una serie de actividades complementarias (excursiones, paseos en barca, etc...).
   Si el cliente está de acuerdo, hará una reserva de viaje en la que figurarán los datos del cliente, el destino, tipo de hotel y tipo de habitación (doble, individual, suite, etc.). El precio del viaje, además de las distintas opciones elegidas, depende de la temporada del destino: alta, baja, etc…"

   Analice la descripción anterior, elabore una lista de especificaciones funcionales y construya un modelo mediante un diagrama Entidad-Relación.

SOLUCIÓN



Observación: El anterior diagrama se ha elaborado con la herramienta DOME. La siguiente tabla resume la equivalencia entre la notación de dicha herramienta y la propuesta en el libro de la asignatura.

Notación libro	Notación DOME
	0:1
	1:1
	0:N
	1:N

INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN

Plan de estudios en extinción
CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

Plan de estudios NUEVO
CÓDIGO CARRERA: 53=SISTEMAS y 54=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

CENTROS PENITENCIARIOS

UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

Departamento de Lenguajes y Sistemas Informáticos

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)
FECHA: junio de 2004 Hora: Duración: 2 horas MATERIAL: NINGUNO

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) la nota mínima es 1 punto; en la segunda parte (ejercicio de teoría aplicada que se valora con 5 puntos) la nota mínima que debe obtener es de 2 puntos.

¡ATENCIÓN! PONGA SUS DATOS EN LA HOJA DE LECTURA ÓPTICA QUE DEBERÁ ENTREGAR JUNTO CON EL RESTO DE LAS RESPUESTAS.

Conteste a las preguntas teóricas, en cualquier orden, en hojas diferentes a las que utilice para la contestación de la segunda parte. En cada parte, la cantidad MÁXIMA de papel (de examen, timbrado) que puede emplear ESTÁ LIMITADA al equivalente a DOS (2) HOJAS de tamaño A4 (210 x 297 mm)

PRIMERA PARTE. PREGUNTAS TEÓRICAS (2'5 PUNTOS CADA UNA)

1. Supóngase que una organización decide afrontar, por primera vez, un proyecto software en el que no tiene experiencia y que se sitúa en un ámbito desconocido para ella. Desde el punto de vista del ciclo de vida, indique qué alternativas tiene esta organización para culminar el proyecto con éxito y explique cómo pueden, dichas alternativas, mitigar los problemas potenciales con los que la organización se puede encontrar durante el desarrollo.

   Reflexión sobre 'Modelos del proceso de desarrollo (Ciclos de Vida)'. En concreto, puntos 1.5 y 1.6 del libro.

2. Defina qué es un TAD (Tipo Abstracto de Datos) y qué repercusiones tiene su uso en el diseño de software.

   Tradicionalmente, en la programación imperativa se optaba por separar los programas en dos partes: la de proceso y la de datos (ejemplos de lenguajes imperativos son FORTRAN, COBOL, PASCAL, BASIC…). La parte de proceso accedía y operaba directamente sobre los datos.
   Esta separación produce una independencia funcional muy baja. Un ejemplo que ilustra esto es el "efecto 2000", donde la simple adición de dígitos al formato de las fechas supuso realizar muchas modificaciones en la parte de proceso.
   Para solventar estos problemas surgió el concepto de Tipo Abstracto de Datos (TAD), que agrupa en una sola entidad la representación de los datos y la parte de proceso que los manipula. Los TADs ocultan la representación de los datos, que sólo es accesible desde las operaciones. De esta forma, un cambio en la representación de los datos de un TAD no se propaga a todo el programa, sino solamente a las operaciones del TAD.
   Tradicionalmente, las estructuras de datos se venían definiendo por su representación. Sin embargo, el paso clave hacia la abstracción de los datos es invertir este punto de vista: olvidar por el momento la representación y considerar que las operaciones en sí mismas definen la estructura de datos.

NOTA: un error grave cometido por muchos alumnos consiste en diseñar una operación fuera del TAD que le corresponde. Por ejemplo, sería erróneo el utilizar un TAD llamado "Rotaciones" y una abstracción funcional, separada del TAD, llamada "ProducirRotaciones".

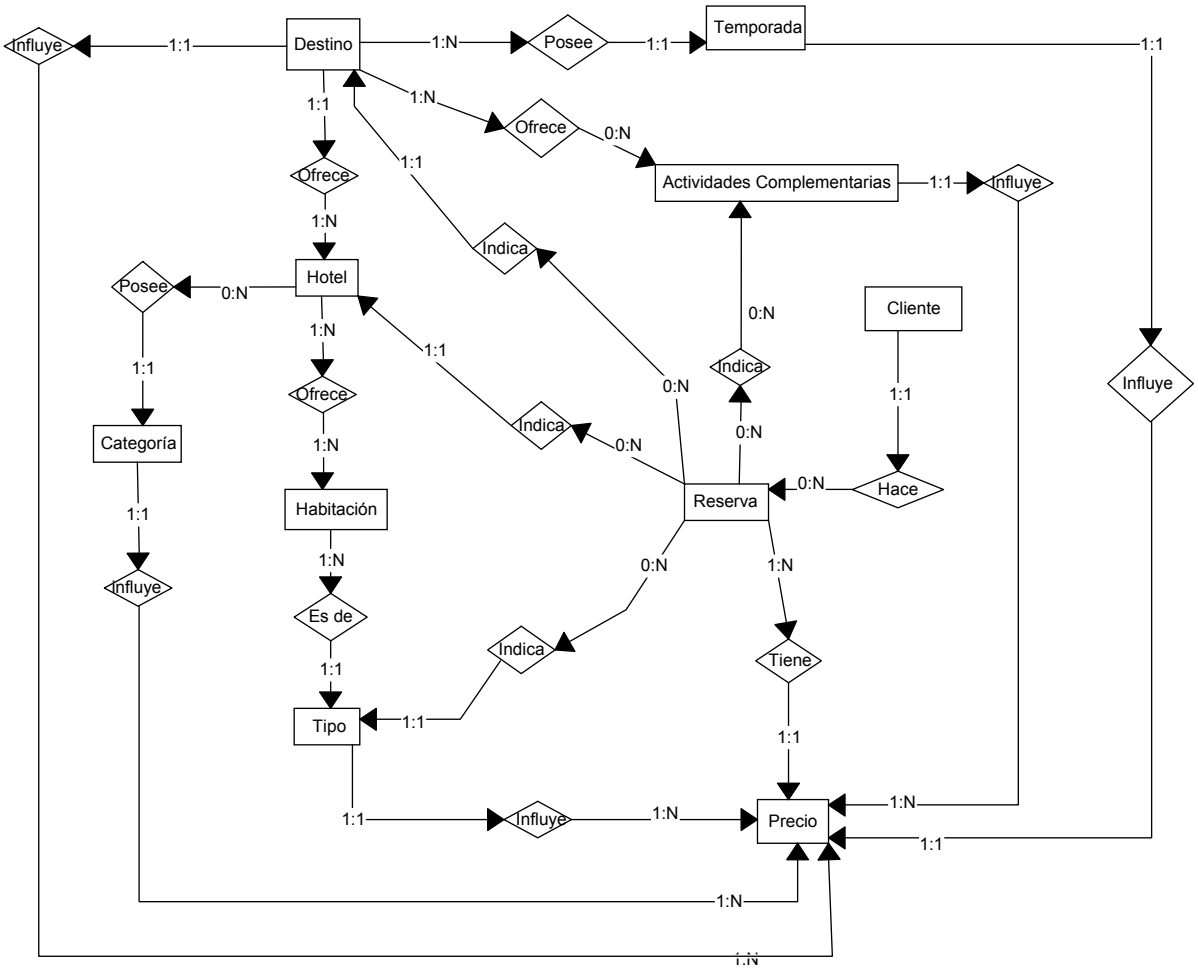
SEGUNDA PARTE. PREGUNTA DE TEORÍA APLICADA (MÁXIMO 5 PUNTOS)

3. Un cliente nos propone desarrollar una aplicación para gestionar algunos aspectos de una agencia de viajes y nos ofrece esta descripción:

   "La agencia de viajes ofrece a sus clientes destinos tanto nacionales como en el extranjero. Para cada destino, la agencia ofrece alojamiento en varios hoteles de distinta categoría. Así mismo, se ofrece, para cada destino, una serie de actividades complementarias (excursiones, paseos en barca, etc...).
   Si el cliente está de acuerdo, hará una reserva de viaje en la que figurarán los datos del cliente, el destino, tipo de hotel y tipo de habitación (doble, individual, suite, etc.). El precio del viaje, además de las distintas opciones elegidas, depende de la temporada del destino: alta, baja, etc…"

   Analice la descripción anterior, elabore una lista de especificaciones funcionales y construya un modelo mediante un diagrama Entidad-Relación.

SOLUCIÓN



Observación: El anterior diagrama se ha elaborado con la herramienta DOME. La siguiente tabla resume la equivalencia entre la notación de dicha herramienta y la propuesta en el libro de la asignatura.

Notación libro	Notación DOME
	0:1
	1:1
	0:N
	1:N

INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN

Plan de estudios en extinción
CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

Plan de estudios NUEVO
CÓDIGO CARRERA: 53=SISTEMAS y 54=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

ORIGINAL NACIONAL

UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

Departamento de Lenguajes y Sistemas Informáticos

ASIGNATURA: **INGENIERÍA DEL SOFTWARE (2º CURSO)**

FECHA: 4 de septiembre de 2004 Hora: 11:30 Duración: 2 horas MATERIAL: NINGUNO

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) la nota mínima es 1 punto; en la segunda parte (ejercicio de teoría aplicada que se valora con 5 puntos) la nota mínima que debe obtener es de 2 puntos.

¡ATENCIÓN! PONGA SUS DATOS EN LA HOJA DE LECTURA ÓPTICA QUE DEBERÁ ENTREGAR JUNTO CON EL RESTO DE LAS RESPUESTAS.

Conteste a las preguntas teóricas, en cualquier orden, en hojas diferentes a las que utilice para la contestación de la segunda parte. En cada parte, la cantidad MÁXIMA de papel (de examen, timbrado) que puede emplear ESTÁ LIMITADA al equivalente a DOS (2) HOJAS de tamaño A4 (210 x 297 mm) (o 4 caras A4, un pliego, un A3 doblado en forma de carpeta, etc.)

PRIMERA PARTE. PREGUNTAS TEÓRICAS (2'5 PUNTOS CADA UNA)

1. Reflexione y explique cómo aplicaría el método de Abbott para analizar un sistema y para construir un modelo con la notación de los DFD.

SOLUCIÓN

La técnica de Abbott establece un procedimiento sistemático para la obtención de elementos significativos de un sistema a partir de una descripción de un modelo. En el caso del diseño, si lo que estamos buscando son abstracciones, partimos de la caracterización o definición de lo que estamos buscando y del espacio de búsqueda (en el libro, una descripción en lenguaje natural) En el diseño se trata de construir una descripción del funcionamiento de la aplicación. Para que dicha aplicación, y su funcionamiento, cumplan con determinados objetivos, se debe construir la descripción descomponiéndola en módulos. Cuando se utilizan abstracciones para el diseño, la descomposición y la descripción vienen caracterizados por ciertos elementos significativos (las abstracciones). Así, al aplicar Abbott en el diseño mediante abstracciones, si lo que se intenta es identificar la existencia de, por ejemplo, una abstracción funcional (la cual no es sino un conjunto de acciones y operaciones que tienen un objetivo concreto); se buscará en la descripción en lenguaje natural el elemento del código lingüístico indica una acción u operación: los verbos. La técnica de Abbott dice: busquemos verbos porque, seguramente, nos ayudarán a localizar las abstracciones funcionales de nuestro sistema.

Entregue la hoja de lectura óptica con sus datos junto con las respuestas de su examen.

El método de Abbott, en realidad, es una forma de análisis. Si se aplica el método descrito anteriormente a la construcción de DFDs, se comprueba su utilidad: ¿qué se busca? ¿procesos? Y ¿qué son los procesos? Actividades y tareas que tendrá que realizar nuestra aplicación. ¿Cuál es el espacio de búsqueda en este caso? La comprensión de lo que tendrá que hacer nuestra aplicación. ¿Cómo representamos en nuestra mente a las actividades y tareas? Mediante verbos o cualquier representación que utilicemos internamente para la idea de acción o tarea. Pues esta es una manera de encontrar los procesos o 'bolas' de un DFD. Se haría igual con los flujos de datos o las entidades externas, etc., sólo que, ahora, buscaremos descripciones de elementos que interactúan con el sistema, sustantivos o adjetivos, etc., que se van a procesar, transformar o almacenar, en el sistema que estamos construyendo.

2. Defina y estructure una clasificación de las técnicas generales para el diseño procedimental del software. Utilice un esquema arbóreo; identifique cada técnica y explique sus características principales. (Hasta aquí la pregunta. La mayor parte de las técnicas de diseño provienen de la programación, lo cual se explica -de forma más o menos implícita- en el desarrollo del libro. Por ello, se valorará positivamente que el estudio anterior se compare y diferencie respecto a otro análogo pero para las técnicas de codificación.)

SOLUCIÓN: (Páginas 160 a 187 del libro. Epígrafes 4.2 a 4.4)

I. Diseño procedimental

1 Diseño funcional descendente

1.1 Técnica del refinamiento progresivo

Aplicación, a la fase de diseño, del concepto de refinamientos sucesivos [Wirth]. Consiste en plantear la aplicación como única operación global, e irla descomponiendo en operaciones más sencillas, detalladas y específicas. En cada nivel de refinamiento, las operaciones identificadas se asignan a módulos separados.

La construcción de programas mediante refinamientos sucesivos proviene de la metodología de programación estructurada [Dijkstra]. Esta metodología de programación consiste en la utilización sólo de estructuras de control claras y sencillas, con esquemas de ejecución con un único punto de inicio y un único punto final.

1.2 Técnica de programación estructurada de Jackson

Aplicación, a la fase de diseño, de la metodología de programación estructurada [Dijkstra]. Esta metodología de programación consiste en la utilización sólo de estructuras de control claras y sencillas, con esquemas de ejecución con un único punto de inicio y un único punto final. Dichas estructuras son la **secuencia**, la **selección** y la **iteración**.

La técnica de Jackson consiste en construir las estructuras del programa de forma similar a las estructuras de los datos de entrada-salida que se manejan.

1.3 Técnica de diseño estructurado [Yourdon]

Técnica de diseño complementaria del 'análisis estructurado' (utilización de DFD). Consiste en:

1. Utilización de DFD para la descripción del funcionamiento de la aplicación.
2. Con los primeros niveles de la descomposición en DFD, se construye un único diagrama en el que se incluyen los procesos implicados y se prescinde de los elementos de

Entregue la hoja de lectura óptica con sus datos junto con las respuestas de su examen.

almacenamiento. Es este último diagrama, se realiza un análisis para identificar o bien un único flujo global (flujo de transformación) o bien uno o varios puntos en los que el flujo global se bifurca (flujo de transacción).

3. Según el patrón identificado en el análisis anterior, se construye el diagrama de diseño mediante la notación de los diagramas de estructura. Para ello, a partir de los DFD iniciales, se asignan procesos o grupos de procesos a los módulos de diseño y se establece una jerarquía o estructura de control en función del mencionado patrón identificado en el punto anterior (transformación o transacción).

2 Diseño con abstracciones

2.1 Descomposición modular con abstracciones

Consiste en dedicar módulos separados a la realización de cada tipo abstracto de datos y cada función importante. Se emplea la notación de los diagramas de bloques jerarquizados; en los que se representan las relaciones de uso y la jerarquía se establece de arriba hacia abajo (‘los de arriba usan a los de abajo’).

Se puede aplicar de forma descendente (ampliación del refinamiento progresivo con las abstracciones) o ascendente (ampliación de primitivas hacia abstracciones de nivel superior).

2.2 Método de Abbott

Metodología para identificar los elementos abstractos que formarán parte del diseño. Las abstracciones obtenidas se complementan y se organizan en un diagrama de diseño. El método se suele aplicar a las descripciones del análisis (formales o informales).

3 Diseño orientado a objetos

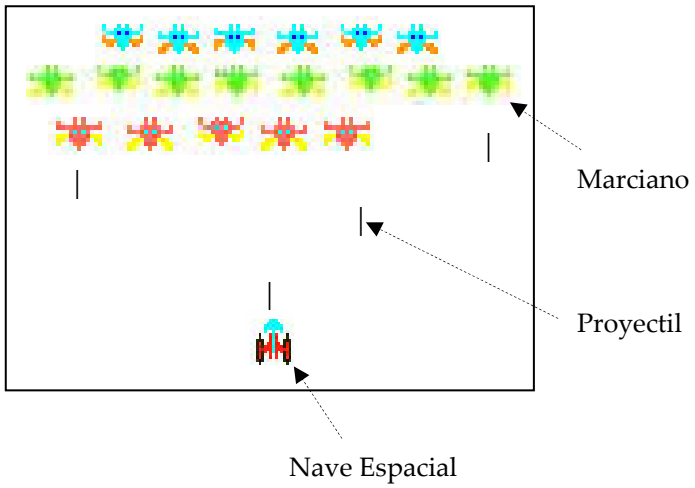
En esencia es similar al diseño con abstracciones pero, además de la relaciones de uso y agregación, hay que considerar la herencia y el polimorfismo. En la descomposición modular del sistema, cada módulo contiene la descripción de una clase o de varias clases de objetos relacionadas entre sí. Se suele utilizar un diagrama de estructura para describir la arquitectura del sistema y las relaciones de uso y diagramas ampliados de las clases y objetos en las que se representan las distintas relaciones entre los datos (herencia, polimorfismo, etc.).

Entregue la hoja de lectura óptica con sus datos junto con las respuestas de su examen.

SEGUNDA PARTE. PREGUNTA DE TEORÍA APLICADA (MÁXIMO 5 PUNTOS)

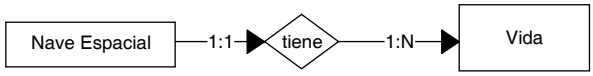
4 Analice mediante un DER (Diagrama Entidad-Relación) la siguiente especificación correspondiente a un videojuego tipo "marcianitos" como el de la figura.

El jugador maneja una nave espacial que dispone inicialmente de cierto número de vidas. La nave debe enfrentarse a distintos tipos de "marcianos" enemigos. Tanto la nave como los marcianos lanzan proyectiles. Cuando la nave recibe un impacto, pierde una de sus vidas. Cuando el nº de vidas de la nave es cero, finaliza la partida. Cada tipo de marciano es capaz de resistir cierto nº de impactos. Algunos marcianos, al ser destruidos desprenden un "bonus"; si la nave captura un bonus, incrementa en uno su número de vidas.

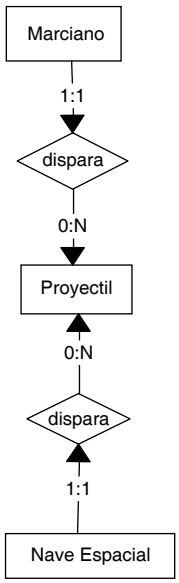


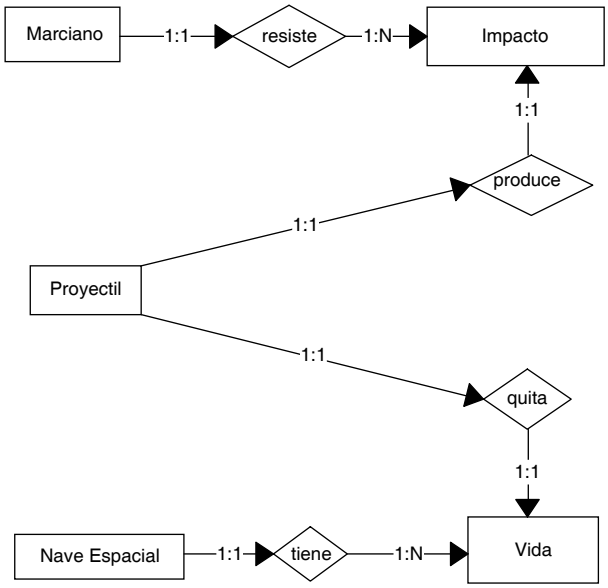
Solución

La construcción del DER solicitado puede abordarse gradualmente. Del texto de la especificación han de extraerse las entidades, las relaciones y la cardinalidad de estas últimas.

El jugador maneja una nave espacial que dispone inicialmente de cierto número de vidas	

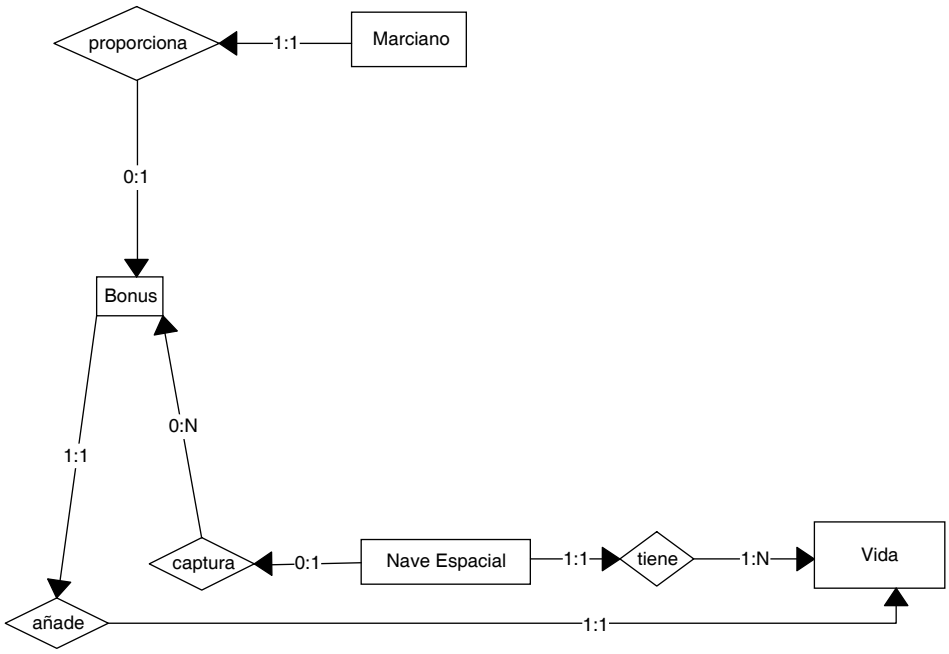
Entregue la hoja de lectura óptica con sus datos junto con las respuestas de su examen.

La nave debe enfrentarse a distintos tipos de "marcianos" enemigos. Tanto la nave como los marcianos lanzan proyectiles.	

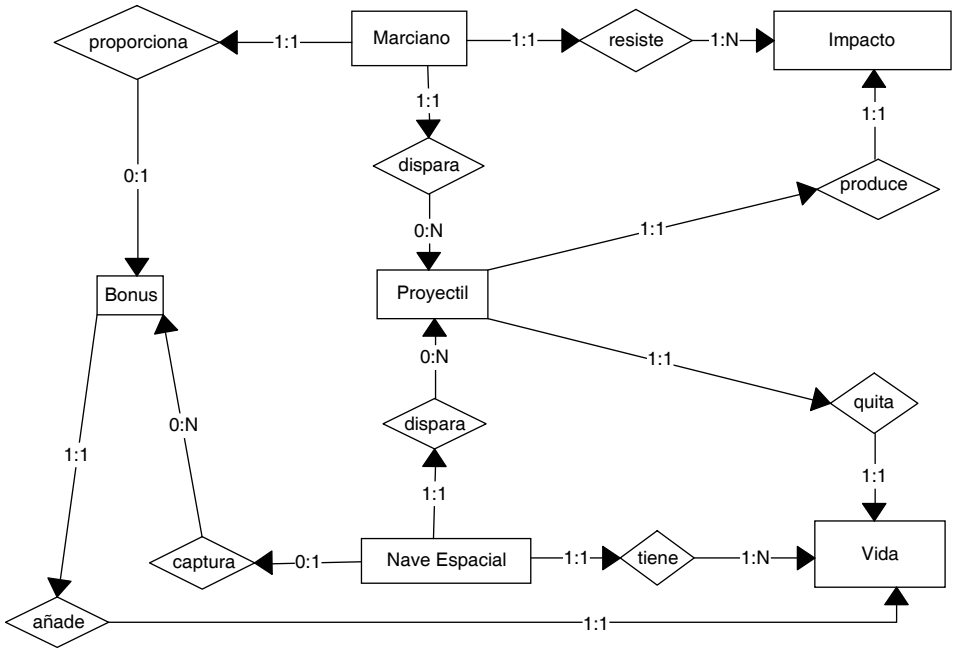
Cuando la nave recibe un impacto, pierde una de sus vidas. Cuando el nº de vidas de la nave es cero, finaliza la partida. Cada tipo de marciano es capaz de resistir cierto nº de impactos.	

Observación: los DER son una notación estática útil para modelar los aspectos estructurales de un a especificación. Los aspectos temporales se pueden modelar con una notación dinámica como los DTE (Diagramas de Transición de Estados). Por eso, la frase del enunciado *Cuando el nº de vidas de la nave es cero, finaliza la partida* no se modela en el DER correspondiente.

Entregue la hoja de lectura óptica con sus datos junto con las respuestas de su examen.

Algunos marcianos, al ser destruidos desprenden un "bonus"; si la nave captura un bonus, incrementa en uno su número de vidas.	

El **resultado final** del análisis es el siguiente DER:



Entregue la hoja de lectura óptica con sus datos junto con las respuestas de su examen.

INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN

Plan de estudios en extinción
CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

Plan de estudios NUEVO
CÓDIGO CARRERA: 53=SISTEMAS y 54=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

NACIONAL RESERVA

Departamento de Lenguajes y Sistemas Informáticos

ASIGNATURA: **INGENIERÍA DEL SOFTWARE (2º CURSO)**

FECHA: 8 de septiembre de 2004 Hora: 9:00 Duración: 2 horas MATERIAL: NINGUNO

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) la nota mínima es 1 punto; en la segunda parte (ejercicio de teoría aplicada que se valora con 5 puntos) la nota mínima que debe obtener es de 2 puntos.

¡ATENCIÓN! PONGA SUS DATOS EN LA HOJA DE LECTURA ÓPTICA QUE DEBERÁ ENTREGAR JUNTO CON EL RESTO DE LAS RESPUESTAS.

Conteste a las preguntas teóricas, en cualquier orden, en hojas diferentes a las que utilice para la contestación de la segunda parte. En cada parte, la cantidad MÁXIMA de papel (de examen, timbrado) que puede emplear ESTÁ LIMITADA al equivalente a DOS (2) HOJAS de tamaño A4 (210 x 297 mm) (o 4 caras A4, un pliego, un A3 doblado en forma de carpeta, etc.)

PRIMERA PARTE. PREGUNTAS TEÓRICAS (2'5 PUNTOS CADA UNA)

1. Explique las propiedades que debe tener el modelo de un sistema software para lograr una especificación correcta.

SOLUCIÓN

Punto 2.2.1 del libro de texto. Páginas 43 a 46.

2. Explique por qué es, la descomposición modular, uno de los objetivos principales del diseño del software.

SOLUCIÓN

Punto 4.1 del libro de texto. Páginas 148 a 160 (resumen).

Entregue la hoja de lectura óptica con sus datos junto con las respuestas de su examen.

SEGUNDA PARTE. PREGUNTA DE TEORÍA APLICADA (MÁXIMO 5 PUNTOS)

3. Se dispone de un Tipo Abstracto de Datos (TAD) *V*, con las operaciones *op1*, *op2 y op3*. Razone cómo se podría derivar un nuevo tipo TAD *N* (utilizando exclusivamente elementos propios de un Diagrama de Abstracciones) que tuviera las operaciones *op1*, *op2*, *op3 y op4*, donde:

   $op1^N$ (*op1* del TAD *N*) sería idéntica a $op1^V$ (*op1* del TAD *V*)

   $op2^N$ tendría un comportamiento distinto a $op2^V$

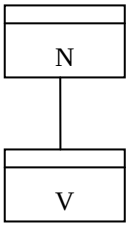
   $op3^N$ sería idéntica a $op3^V$

   $op4^N$ sería totalmente nueva

   ¿Cuál sería la solución si *V* y *N* fueran Clases de Objetos y se aplicaran Herencia y Polimorfismo?

SOLUCIÓN

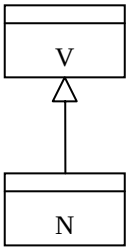
a) *N* y *V* como TADs



Lista de operaciones de *N*:

Operación	Pseudocódigo
$op1^N$	llamar a $op1^V$
$op2^N$	<<nuevo código>>
$op3^N$	llamar a $op3^V$
$op4^N$	<<nuevo código>>

b) *N* y *V* como Clases de Objetos



Lista de operaciones de *N*:

Operación	Pseudocódigo
$op2^N$	<<nuevo código>>
$op4^N$	<<nuevo código>>

Como *N* hereda de *V*, *N* adquiere automáticamente las operaciones de *V*. Por lo tanto, no sería necesario reescribir las operaciones *op1* y *op3* en N. En cambio, sí habría que añadir $op2^N$, que sería invocada adecuadamente gracias al polimorfismo de anulación. Finalmente, también habría que añadir a N la nueva operación $op4^N$.

Entregue la hoja de lectura óptica con sus datos junto con las respuestas de su examen.

INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN

Plan de estudios en extinción
CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

Plan de estudios NUEVO
CÓDIGO CARRERA: 53=SISTEMAS y 54=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

ORIGINAL EXTRANJERO

UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

Departamento de Lenguajes y Sistemas Informáticos

ASIGNATURA: **INGENIERÍA DEL SOFTWARE (2º CURSO)**

FECHA: 4 de septiembre de 2004 Hora: Duración: 2 horas MATERIAL: NINGUNO

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) la nota mínima es 1 punto; en la segunda parte (ejercicio de teoría aplicada que se valora con 5 puntos) la nota mínima que debe obtener es de 2 puntos.

¡ATENCIÓN! PONGA SUS DATOS EN LA HOJA DE LECTURA ÓPTICA QUE DEBERÁ ENTREGAR JUNTO CON EL RESTO DE LAS RESPUESTAS.

Conteste a las preguntas teóricas, en cualquier orden, en hojas diferentes a las que utilice para la contestación de la segunda parte. En cada parte, la cantidad MÁXIMA de papel (de examen, timbrado) que puede emplear ESTÁ LIMITADA al equivalente a DOS (2) HOJAS de tamaño A4 (210 x 297 mm) (o 4 caras A4, un pliego, un A3 doblado en forma de carpeta, etc.)

PRIMERA PARTE. PREGUNTAS TEÓRICAS (2'5 PUNTOS CADA UNA)

1. Explique las ventajas del uso de prototipos, en sus distintas variantes, durante el ciclo de vida.

SOLUCIÓN

Punto 1.5 del libro de texto. Páginas 16, 17 y, parcialmente, las siguientes de este punto.

2. Defina y estructure una clasificación de las técnicas generales para el diseño procedimental del software. Utilice un esquema arbóreo; identifique cada técnica y explique sus características principales. (Hasta aquí la pregunta. La mayor parte de las técnicas de diseño provienen de la programación, lo cual se explica -de forma más o menos implícita- en el desarrollo del libro. Por ello, se valorará positivamente que el estudio anterior se compare y diferencie respecto a otro análogo pero para las técnicas de codificación.)

SOLUCIÓN

Páginas 160 a 187 del libro. Epígrafes 4.2 a 4.4

Entregue la hoja de lectura óptica con sus datos junto con las respuestas de su examen.

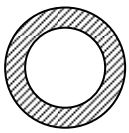
SEGUNDA PARTE. PREGUNTA DE TEORÍA APLICADA (MÁXIMO 5 PUNTOS)

3. Se desea construir un programa informático para el dibujo de figuras geométricas planas. La siguiente tabla resume las figuras que manejará el programa y los elementos necesarios para la determinación de cada una de ellos.

Figura	Datos que la determinan
Punto	Dos coordenadas cartesianas
Recta	Dos puntos
Rectángulo	Dos puntos (que determinan su diagonal)
Círculo	Un punto (su centro) y su radio (un número real que representa una longitud)

a) Realice un diseño mediante un Diagrama Orientado a Objetos que represente las figuras anteriores y sus relaciones.

b) Se desea añadir al programa anterior el manejo de la figura "corona circular" (ver figura). Indique cómo puede obtenerse esta nueva figura aprovechando el desarrollo anterior mediante:
   I. Herencia
   II. Composición



Corona Circular

SOLUCIÓN

Parte a)
De nuestro enunciado se desprende rápidamente que vamos a tener las siguientes clases:

CLASE Punto
ATRIBUTOS
coordenadaX : Tipo numérico
coordenadaY : Tipo numérico
OPERACIONES
....
FIN-CLASE

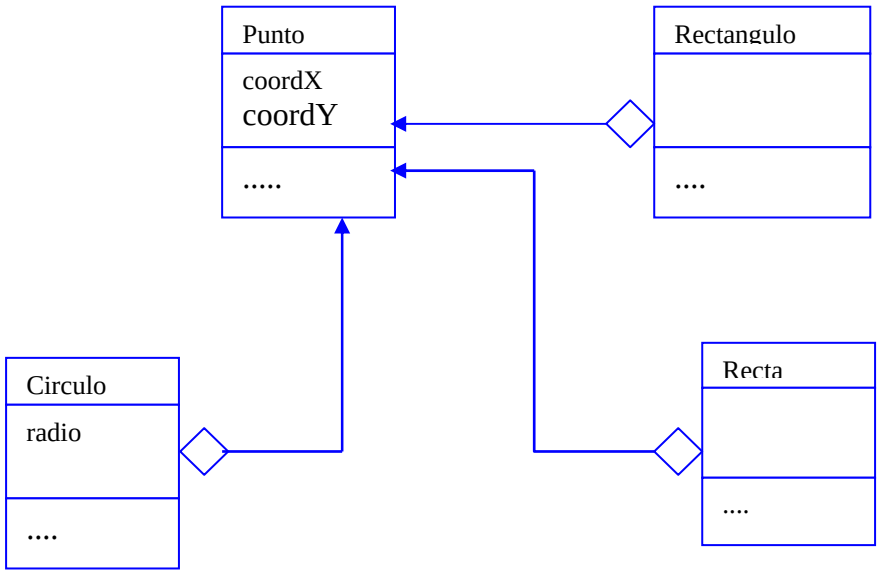
CLASE Rectangulo
ATRIBUTOS
derechaSup : Tipo Punto
izquierdaInf : Tipo Punto
OPERACIONES
....
FIN-CLASE

CLASE Recta
ATRIBUTOS
punto1 : Tipo Punto
punto2 : Tipo Punto
OPERACIONES
...
FIN-CLASE

CLASE Circulo
ATRIBUTOS
radio : Tipo numérico
centro : Tipo Punto
OPERACIONES
. ...
FIN-CLASE

Entregue la hoja de lectura óptica con sus datos junto con las respuestas de su examen.

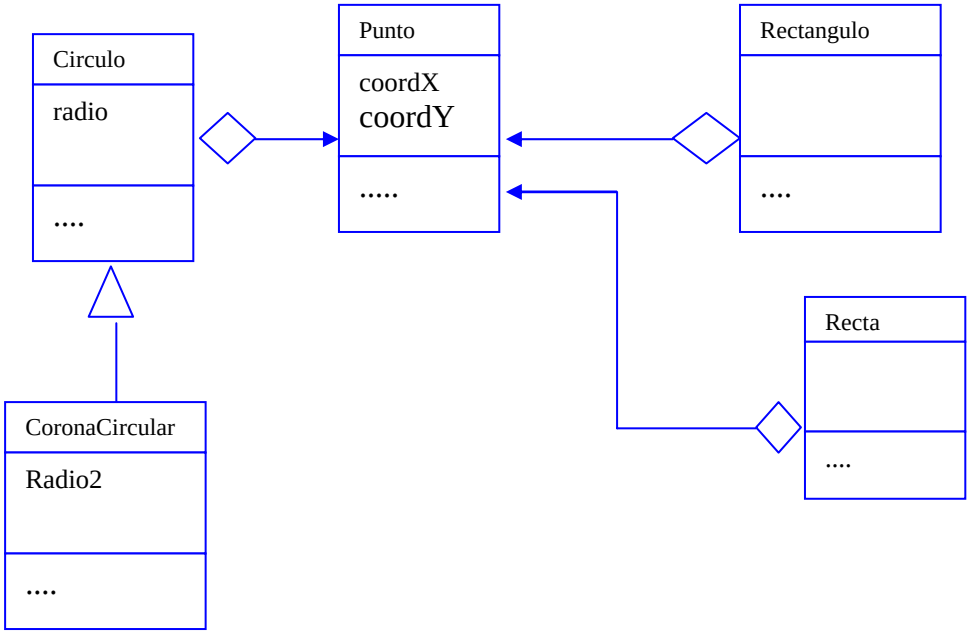
Y el diagrama de clases que representa esto sería:



Parte b)

Solución I.

Podemos pensar que una *corona circular* **ES UN** *circulo* con dos radios, esto es, la nueva clase CoronaCircular puede aprovechar todas las características de la clase Circulo, añadiendo a su vez un nuevo radio, quedando perfectamente determinada toas las características de la corona circular. Todas las propiedades de la clase círculo las obtenemos por tanto mediante herencia, pero además, añadiremos un atributo tipo numérico que representa el segundo radio. El siguiente diagrama representa esta situación:

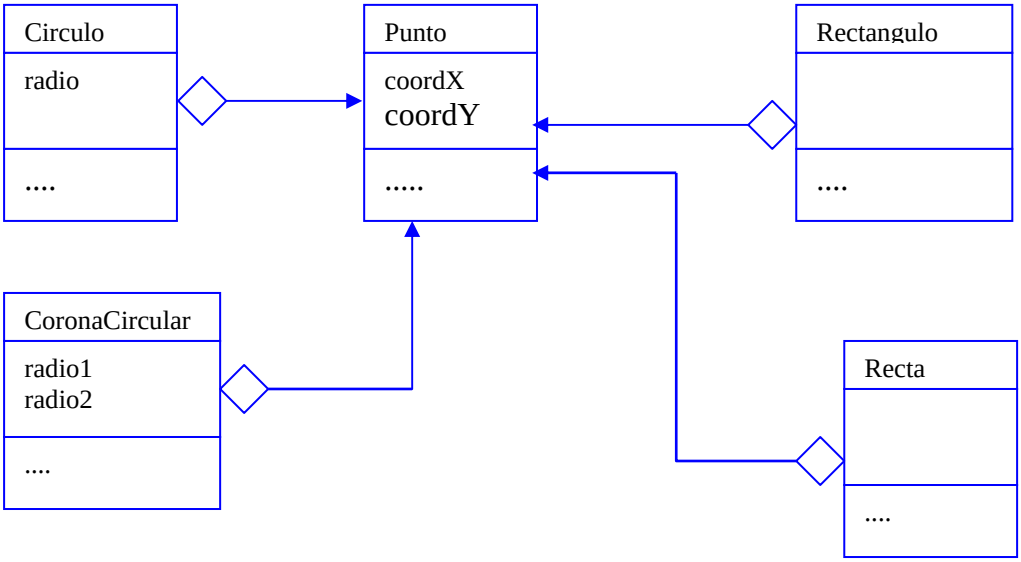


Entregue la hoja de lectura óptica con sus datos junto con las respuestas de su examen.

Solución II.

Por otra parte, podemos ’visualizar’ una corona circular como la diferencia de dos círculos. Por tanto, la clase CoronaCircular2 será una clase que contenga dos atributos (en su interior) de tipo Circulo, esto es, la CoronaCircular2 está compuesta de dos círculos.

La figura siguiente refleja la solución II:



Entregue la hoja de lectura óptica con sus datos junto con las respuestas de su examen.

INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN

Plan de estudios en extinción
CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

Plan de estudios NUEVO
CÓDIGO CARRERA: 53=SISTEMAS y 54=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

RESERVA EXTRANJERO

UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

Departamento de Lenguajes y Sistemas Informáticos

ASIGNATURA: **INGENIERÍA DEL SOFTWARE (2º CURSO)**

FECHA: 8 de septiembre de 2004 Hora: Duración: 2 horas MATERIAL: NINGUNO

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) la nota mínima es 1 punto; en la segunda parte (ejercicio de teoría aplicada que se valora con 5 puntos) la nota mínima que debe obtener es de 2 puntos.

¡ATENCIÓN! PONGA SUS DATOS EN LA HOJA DE LECTURA ÓPTICA QUE DEBERÁ ENTREGAR JUNTO CON EL RESTO DE LAS RESPUESTAS.

Conteste a las preguntas teóricas, en cualquier orden, en hojas diferentes a las que utilice para la contestación de la segunda parte. En cada parte, la cantidad MÁXIMA de papel (de examen, timbrado) que puede emplear ESTÁ LIMITADA al equivalente a DOS (2) HOJAS de tamaño A4 (210 x 297 mm) (o 4 caras A4, un pliego, un A3 doblado en forma de carpeta, etc.)

PRIMERA PARTE. PREGUNTAS TEÓRICAS (2'5 PUNTOS CADA UNA)

1. Enuncie, defina y explique los tres tipos de mantenimiento que se estudian en esta asignatura.

SOLUCIÓN

Punto 1.8.1 del libro de texto. Página 24.

2. Reflexione y explique cómo aplicaría el método de Abbott para analizar un sistema y para construir un modelo con la notación de los DFD.

SOLUCIÓN

La técnica de Abbott establece un procedimiento sistemático para la obtención de elementos significativos de un sistema a partir de una descripción de un modelo. En el caso del diseño, si lo que estamos buscando son abstracciones, partimos de la caracterización o definición de lo que estamos buscando y del espacio de búsqueda (en el libro, una descripción en lenguaje natural) En el diseño se trata de construir una descripción del funcionamiento de la aplicación. Para que dicha aplicación, y su funcionamiento, cumplan con determinados objetivos, se debe construir la descripción descomponiéndola en módulos. Cuando se utilizan abstracciones para el diseño, la descomposición y la descripción vienen caracterizados por ciertos elementos significativos (las abstracciones). Así, al aplicar Abbott en el diseño mediante abstracciones, si lo que se intenta es identificar la existencia de, por ejemplo, una abstracción funcional (la cual no es sino un conjunto de acciones y operaciones que tienen un objetivo concreto); se buscará en la descripción en lenguaje natural el elemento del código lingüístico indica una acción u operación: los verbos. La técnica de Abbott dice: busquemos verbos porque, seguramente, nos ayudarán a localizar las abstracciones funcionales de nuestro sistema.

Entregue la hoja de lectura óptica con sus datos junto con las respuestas de su examen.

El método de Abbott, en realidad, es una forma de análisis. Si se aplica el método descrito anteriormente a la construcción de DFDs, se comprueba su utilidad: ¿qué se busca? ¿procesos? Y ¿qué son los procesos? Actividades y tareas que tendrá que realizar nuestra aplicación. ¿Cuál es el espacio de búsqueda en este caso? La comprensión de lo que tendrá que hacer nuestra aplicación. ¿Cómo representamos en nuestra mente a las actividades y tareas? Mediante verbos o cualquier representación que utilicemos internamente para la idea de acción o tarea. Pues esta es una manera de encontrar los procesos o 'bolas' de un DFD. Se haría igual con los flujos de datos o las entidades externas, etc., sólo que, ahora, buscaremos descripciones de elementos que interactúan con el sistema, sustantivos o adjetivos, etc., que se van a procesar, transformar o almacenar, en el sistema que estamos construyendo.

SEGUNDA PARTE. PREGUNTA DE TEORÍA APLICADA (MÁXIMO 5 PUNTOS)

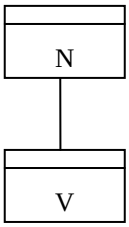
3. Se dispone de un Tipo Abstracto de Datos (TAD) *V*, con las operaciones *op1*, *op2 y op3*. Razone cómo se podría derivar un nuevo tipo TAD *N* (utilizando exclusivamente elementos propios de un Diagrama de Abstracciones) que tuviera las operaciones *op1*, *op2*, *op3 y op4*, donde:

   $op1^N$ (*op1* del TAD *N*) sería idéntica a $op1^V$ (*op1* del TAD *V*)
   $op2^N$ tendría un comportamiento distinto a $op2^V$
   $op3^N$ sería idéntica a $op3^V$
   $op4^N$ sería totalmente nueva

   ¿Cuál sería la solución si *V* y *N* fueran Clases de Objetos y se aplicaran Herencia y Polimorfismo?

SOLUCIÓN

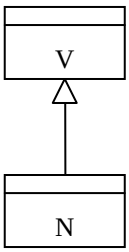
a) *N* y *V* como TADs



Lista de operaciones de *N*:

Operación	Pseudocódigo
$op1^N$	llamar a $op1^V$
$op2^N$	<<nuevo código>>
$op3^N$	llamar a $op3^V$
$op4^N$	<<nuevo código>>

b) *N* y *V* como Clases de Objetos



Lista de operaciones de *N*:

Operación	Pseudocódigo
$op2^N$	<<nuevo código>>
$op4^N$	<<nuevo código>>

Como *N* hereda de *V*, *N* adquiere automáticamente las operaciones de *V*. Por lo tanto, no sería necesario reescribir las operaciones *op1* y *op3* en N. En cambio, sí habría que añadir $op2^N$, que sería invocada adecuadamente gracias al polimorfismo de anulación. Finalmente, también habría que añadir a N la nueva operación $op4^N$.

Entregue la hoja de lectura óptica con sus datos junto con las respuestas de su examen.