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

CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

CÓDIGO CARRERA:
Plan de estudios en extinción: 40=SISTEMAS y 41=GESTIÓN
Plan de estudios NUEVO: 53=SISTEMAS y 54=GESTIÓN

MATERIAL PERMITIDO: NINGUNO

MODELO: NACIONAL 1ª SEMANA

Departamento de Ingeniería de Software y Sistemas Informáticos

---

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En 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. ¿Qué es la línea base? ¿Para qué sirve? ¿Cómo se usa?

SOLUCIÓN

Epígrafe 1.9.5 "Gestión de la configuración", '*Control de cambios*', los dos últimos párrafos de la página 31 hasta el final del epígrafe en la página 32.

2. ¿Por qué se busca que la descomposición modular de un diseño cumpla con las cualidades de "Independencia Funcional", "Comprensibilidad" y "Adaptabilidad"?

SOLUCIÓN

Además de conseguir el objetivo fundamental del diseño, que es la descripción/especificación adecuada (es decir, que permita la implementación en un lenguaje) del funcionamiento de la aplicación; dicho diseño debe permitir un mantenimiento sencillo y un desarrollo que economice recursos. Para ello se hace la 'descomposición modular'. En las páginas 150 a 160, se encuentra la justificación de en qué medida, las tres cualidades del enunciado, contribuyen a conseguir ese mantenimiento sencillo e implementación con ahorro de recursos.

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

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

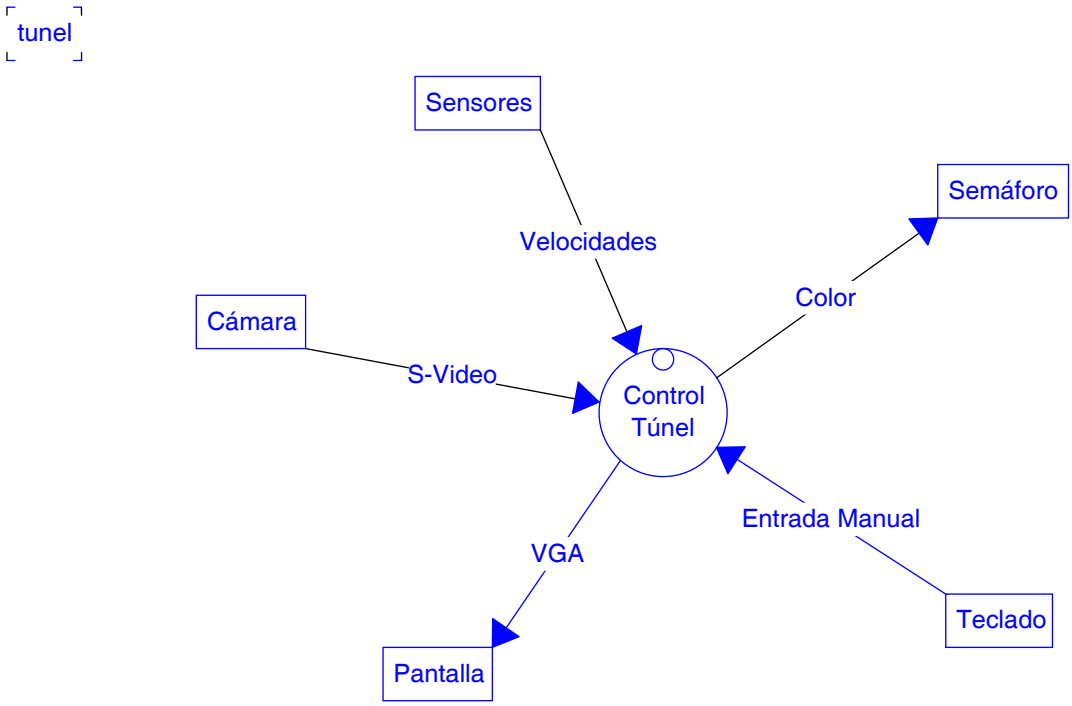
3. Se desea realizar una aplicación informática para evitar congestiones de automóviles en el interior de un túnel. El sistema recibe datos de la velocidad de los automóviles provenientes de un conjunto de sensores distribuidos en el interior del túnel. Los datos de todos los sensores se reciben en paralelo y de forma periódica. El sistema deberá actuar, en función de dichos datos, sobre un semáforo situado a la entrada del túnel, de forma que cuando se detecte un coche circulando a una velocidad inferior a una estipulada, el semáforo se pondrá en rojo y cuando las velocidades detectadas por los sensores sean superiores a otro valor estipulado, se volverá a poner en verde. Existirá un centro de control con una pantalla que muestre el estado del semáforo y las velocidades instantáneas captadas por el conjunto de sensores. El sistema podrá funcionar en modo manual, para lo cual el operario deberá introducir por teclado su identificador y contraseña cada vez que quiera pasar de automático a manual, o viceversa, y/o cambiar el estado del semáforo. Una o varias cámaras de video enviarán imágenes del interior y exterior del túnel, que se mostrarán en la pantalla y servirán para ayudar al operario a tomar decisiones cuando realice el control manual. El sistema almacenará todos los datos sobre velocidades proporcionadas por los sensores, el estado del semáforo, los cambios de estado que se produzcan en él y su origen (automático o manual) y los accesos de los usuarios, junto con la fecha y hora de cada evento.

   Se pide: analizar el sistema mediante Diagramas de Flujo de Datos (DFDs), desarrollando los DFDs de contexto, nivel 0 y 1.

SOLUCIÓN

Errata: DFD de contexto = DFDs de nivel 0. Evidentemente, lo que se pedía es '**nivel 0 y 1**' o bien '**DFD de contexto y nivel 1**'.

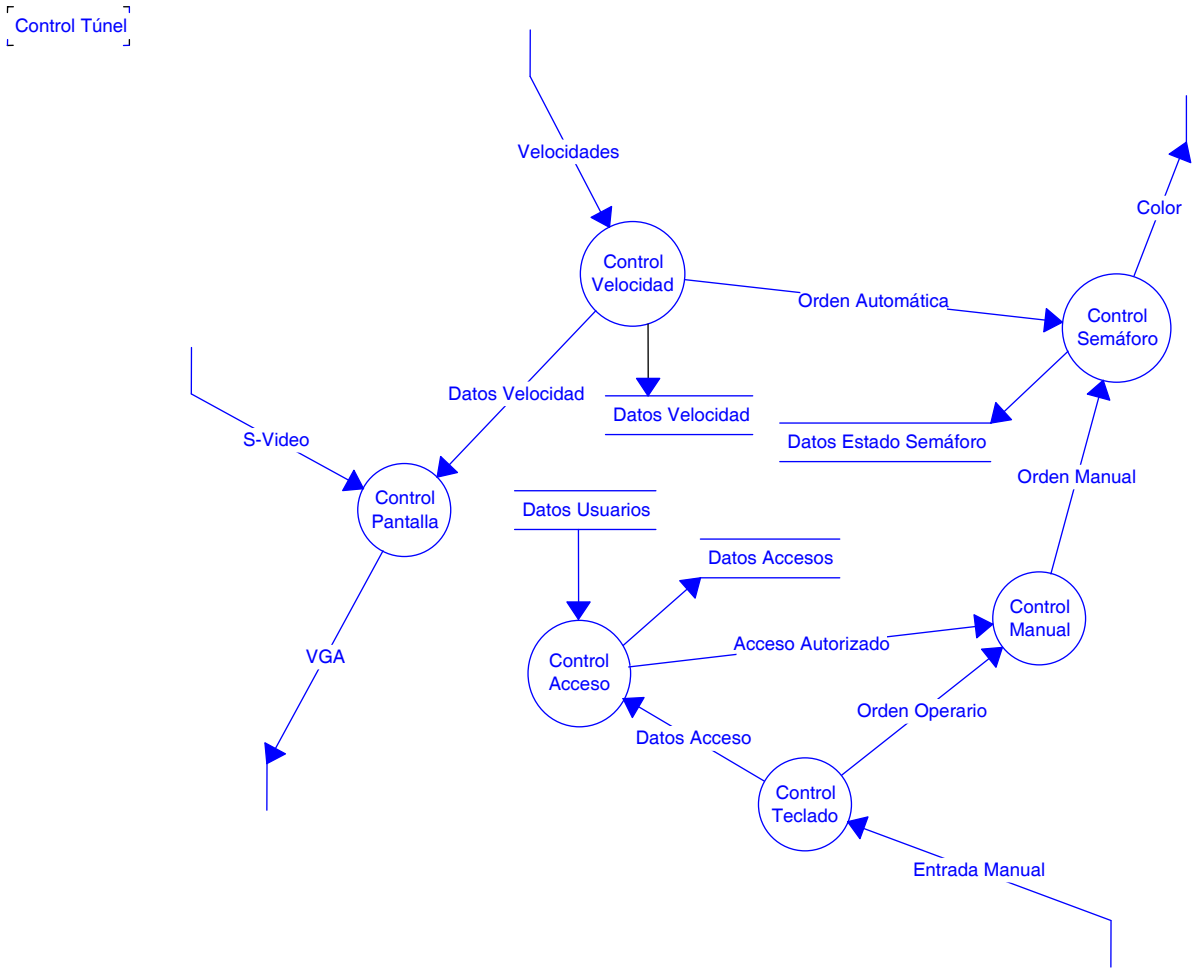
DFD de contexto:



Aplicación informática para evitar congestiones de automóviles en el interior de un túnel. El sistema recibe datos de la velocidad de los automóviles provenientes de un conjunto de sensores distribuidos en el interior del túnel. Los datos de todos los sensores se reciben en paralelo y de forma periódica. El sistema deberá actuar, en función de dichos datos, sobre un semáforo situado a

la entrada del túnel, de forma que cuando se detecte un coche circulando a una velocidad inferior a una estipulada, el semáforo se pondrá en rojo y cuando las velocidades detectadas por los sensores sean superiores a otro valor estipulado, se volverá a poner en verde.
Existirá un centro de control con una pantalla que muestre el estado del semáforo y las velocidades instantáneas captadas por el conjunto de sensores. El sistema podrá funcionar en modo manual, para lo cual el operario deberá introducir por teclado su identificador y contraseña cada vez que quiera pasar de automático a manual, o viceversa, y/o cambiar el estado del semáforo. Una o varias cámaras de video enviarán imágenes del interior y exterior del túnel, que se mostrarán en la pantalla y servirán para ayudar al operario a tomar decisiones cuando realice el control manual. El sistema almacenará todos los datos sobre velocidades proporcionadas por los sensores, el estado del semáforo, los cambios de estado que se produzcan en él y su origen (automático o manual) y los accesos de los usuarios, junto con la fecha y hora de cada evento.

DFD de nivel 1:



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

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

CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

CÓDIGO CARRERA:
Plan de estudios en extinción: 40=SISTEMAS y 41=GESTIÓN
Plan de estudios NUEVO: 53=SISTEMAS y 54=GESTIÓN

MATERIAL PERMITIDO: NINGUNO

MODELO: NACIONAL 2ª SEMANA

Departamento de Ingeniería de Software y Sistemas Informáticos

---

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En 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. La notación de los DFD ¿a qué metodología de análisis está asociada? ¿Cómo se construye un modelo mediante esta notación? ¿Se le ocurre algún procedimiento para identificar los "procesos" y los "flujos de datos" del sistema?

SOLUCIÓN

La notación de los DFD está asociada al análisis estructurado (primer párrafo del epígrafe 4.2.3 "Diseño estructurado", página 165).

Una descripción completa de cómo se construye el modelo, mediante esta notación, aparece en las páginas 52 a 57, epígrafe 2.3.2 "Diagramas de flujo de datos".

Para identificar "procesos" y "flujos de datos" se podría utilizar Abbott (página 173), buscando identificar la información, de distinta naturaleza, que 'fluye' en distintas direcciones (flujos) y las descripciones de las acciones por las cuales se transforman esos flujos de información (los procesos).

2. Defina qué es un Tipo Abstracto de Datos (TAD) y deduzca las relaciones que se puedan establecer con los conceptos de "ocultación", "genericidad", "herencia" y "polimorfismo".

SOLUCIÓN

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.

Por tanto, la "ocultación" está garantizada en el TAD, gracias al propio concepto de esta abstracción que oculta la representación de los datos y sólo deja visible las operaciones que manipulan a dichos datos.

Por la propia naturaleza del TAD, se pueden definir tipos de datos que recogen una solución "genérica" o aplicable a varios casos particulares. Un tipo 'genérico' es un tipo parametrizable, es decir, que cubre un abanico de soluciones particulares y se adecua a distintos usos.

En cuanto a la "herencia", un TAD es una abstracción y no está dotada del mecanismo de la herencia.

Por último, la capacidad representar algo o de actuar de distinta forma, sin que se pierda la propia naturaleza (polimorfismo), se puede alcanzar mediante la genericidad. Por tanto, un TAD puede ser polimorfo si es genérico; no es así con respecto al polimorfismo de herencia, del que no dispone un TAD por ser una abstracción.

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

3. Como resultado del diseño arquitectónico de una aplicación hemos caracterizado un módulo llamado 'Gestión de fechas', que realiza las siguientes operaciones sobre una lista de fechas:
   - Buscar la más reciente.
   - Buscar la más antigua.
   - Imprimir todas las fechas que pertenezcan a cierto mes.

Para este módulo, proponga un diseño mediante refinamiento progresivo utilizando:

A. Funciones.

B. Abstracciones.

SOLUCIÓN

A. Tradicionalmente, la atención en el desarrollo del software se centraba en los aspectos funcionales, es decir, se trataba de responder a la cuestión ¿Cómo debe funcionar el programa para comportarse según lo acordado en el análisis? Entre las técnicas o metodologías de diseño en esta época se encuentra la descomposición funcional por refinamientos progresivos.

La descomposición funcional por refinamientos progresivos es una técnica de diseño que proviene de la metodología de programación, desarrollada en los años 60 y 70 principalmente por Dijkstra, y se conoce como programación estructurada. Su principal característica consiste en ver la aplicación compuesta de funciones o procedimientos donde entran unos datos y salen otros transformados. Para lograr esto de forma eficiente y estructurada se diseña atendiendo a las siguientes directrices:

- La especificación de las funciones se expresa mediante un conjunto de construcciones lógicas, las cuales deben garantizar la realización de cualquier tarea y deben ser suficientemente formales para evitar ambigüedades. Para ello se propone como constructores la *secuencia*, la *condición* y la *repetición* empleando para ello un lenguaje en pseudocódigo.
- Se aplica el refinamiento progresivo para el desarrollo de la aplicación, en el siguiente sentido: se plantea inicialmente el programa como una única función global y se va descomponiendo (refinando) poco a poco en funciones más sencillas.
- Las estructuras de datos, importantes en cualquier diseño, deben estar supeditadas a la funcionalidad de la aplicación. Estas funciones, utilizan los datos como elementos de

entrada para su utilización o trasformación; esto se puede conseguir haciendo que los datos sean variables globales (son vistas desde cualquier función, aunque se pierda cohesión) o se pasan como parámetros de entrada. Estas estructuras de datos se pueden diseñar en cualquier momento y con independencia de la parte operacional.

Primero empezamos planteando el módulo Gestión de Fechas como una función global e iremos descomponiéndola es otras funciones. Para especificar el comportamiento de estas funciones utilizamos pseudocódigo:

```
Gestión de fechas →
     CASO opcion de menu
       SI_ES fecha reciente ENTONCES BuscarFechaReciente
       SI_ES fecha antigua ENTONCES BuscarFechaAntigua
       SI_ES listar fecha del mes ENTONCES ImprimirFechaMes

BuscarFechaReciente →
     FechaInicial = Primer elemento de la Lista de Fechas
       PARA_CADA Elemento EN Lista de Fechas HACER
        SI FechaInicial es MENOR que Elemento ENTONCES
          FechaInicial = Elemento
        FIN-SI
       FIN-PARA

              ...
```

B. Posteriormente, se comenzó a otorgar mayor relevancia a los datos. Es decir, el desarrollo de una aplicación pasó a enfocarse como: ¿qué conceptos o datos intervienen en el programa?, ¿cuáles son sus operaciones o responsabilidades? y ¿cómo se relacionan entre sí?

En un diseño basado en abstracciones, antes de realizar la descomposición funcional u operativa de la aplicación deben quedar identificadas las posibles abstracciones (tipos abstractos de datos, funciones o bien datos encapsulados) que aparecerán en nuestro diseño. Para ello se siguen los siguientes pasos:

i. Identificar claramente las abstracciones que aparezcan en la aplicación. Esto se puede hacer, a partir de las especificaciones, utilizando, por ejemplo, el método de búsqueda de Abbott.
ii. Definir las operaciones asociadas a cada abstracción.
iii. Describir las operaciones mediante lenguaje natural.
iv. A partir de las especificaciones del programa llevar a cabo una descomposición funcional como la descrita en el apartado anterior. Así mismo reemplazar las descripciones de las operaciones de cada abstracción mediante construcciones estructuradas.

Podemos ver el módulo Gestión de fechas como una abstracción funcional global que gestiona a todas las demás abstracciones.

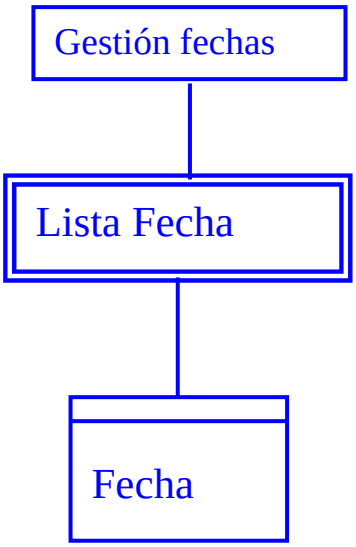
Podemos identificar la lista de fechas como un tipo abstracto de datos, ahora bien, como sólo vamos a tener una lista, podríamos considerarlo un tipo encapsulado. A esta lista estarían asociadas las operaciones de buscar fechas. Otro tipo de datos abstractos sería el que encapsularía tipo fecha y podríamos asociarle, entre otras operaciones, la de comparar con otra fecha.

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

Dato: Lista fecha (Dato encapsulado)
  Atributos
    lista de elementos del tipo fecha
  Operaciones
    Buscar fecha reciente
    Buscar fecha antigua
    Listar fecha mes

Dato: Fecha (Tipo Abstracto de datos)
  Atributos
    Año
    Mes
    Día
  Operaciones
    Comparar con otra fecha

DIAGRAMA DE ABSTRACCIONES



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

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

CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

CÓDIGO CARRERA:
Plan de estudios en extinción: 40=SISTEMAS y 41=GESTIÓN
Plan de estudios NUEVO: 53=SISTEMAS y 54=GESTIÓN

MATERIAL PERMITIDO: NINGUNO

MODELO: EXTRANJERO ORIGINAL
(Europa Continental y Guinea)

Departamento de Ingeniería de Software y Sistemas Informáticos

---

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En 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)

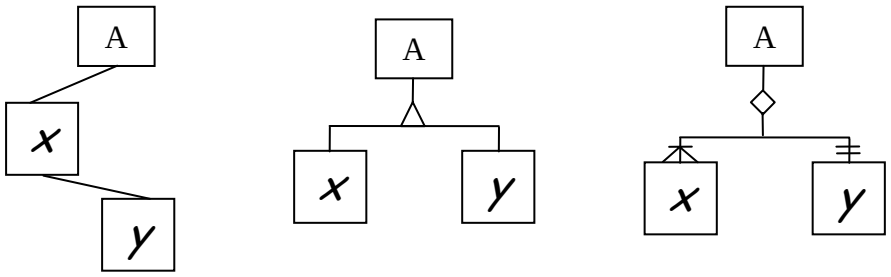
PRIMERA PARTE. PREGUNTAS TEÓRICAS (2'5 PUNTOS CADA UNA)

1. ¿Qué punto de vista ofrecen los diagramas de transición de estados en la comprensión del sistema?

SOLUCIÓN

La evolución temporal del comportamiento deseado para el sistema. Epígrafe 2.3.3, primer y segundo párrafo del epígrafe, en las páginas 57 y 58.

2. Enumere y explique las relaciones entre bloques que se representan en estos diagramas de diseño. ¿Se puede producir 'sobrecarga' en el módulo 'x' del diagrama 'C)'?



A) B) C)

SOLUCIÓN

La notación corresponde a 'diagramas de estructura'; utilizados como notación de diseño, principalmente, en el diseño arquitectónico.

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

En el caso A), se representan relaciones de uso: ‘A’ llama o utiliza el módulo ‘x’ y, éste, invoca o utiliza el módulo ‘y’.

En el caso B), se representan relaciones de herencia (que sólo se dan en los objetos): ‘A’ es el antecedente o ‘padre’ de los módulos ‘x’ e ‘y’, que son sus hijos. Por tanto, heredarán las propiedades estáticas y dinámicas del ‘padre’. Es en los hijos donde se puede producir *especialización*, adecuando las propiedades heredadas al uso especializado del ‘hijo’ o incorporando nuevas.

En el caso C), se representan relaciones de composición o agregación: ‘A’ está compuesto por una cantidad (se representa la cardinalidad de la relación) de instancias del módulo ‘x’ y por un número de instancias del módulo ‘y’. La *sobrecarga* es una manera de expresar el polimorfismo que está vinculado, estrechamente, con el mecanismo de la herencia y sólo se produce con la parte dinámica de los objetos (‘metodos’). Como lo que se representa en el diagrama es la composición y no la herencia, no se puede decir si existe la posibilidad de sobrecarga en este módulo concreto. Podría producirse sobrecarga, si el módulo ‘x’ fuera una clase u objeto que heredara de otro; en cuyo caso, aparecería en otro diagrama distinto en el que se representaría la herencia.

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

3. Dado el subprograma `Factorial` escrito en Modula 2, que calcula el factorial de un número entero positivo y devuelve el valor **-1** en caso de ser negativo, verifique su funcionamiento correcto con pruebas:
   - De caja negra: determine las clases de equivalencia necesarias e implemente un subprograma llamado `TestsCajaNegra` que ejecute las pruebas correspondientes.
   - De caja transparente: implemente un subprograma llamado `TestsCajaTransparente` que pruebe el cubrimiento lógico y el bucle (cuyo número de repeticiones no está acotado).

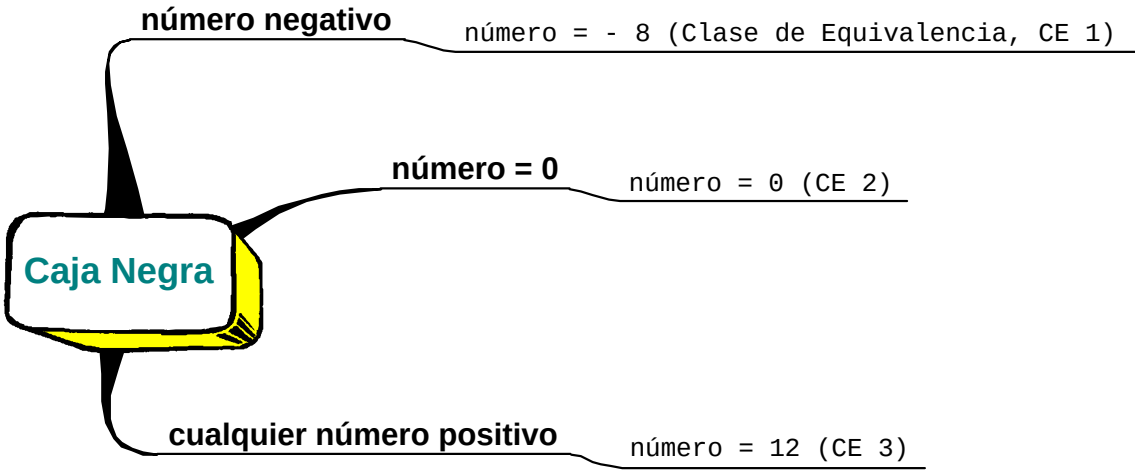
   Nota: la codificación de los subprogramas pedidos puede realizarse en pseudocódigo o en Modula-2.

```
PROCEDURE Factorial(numero: INTEGER): INTEGER;
VAR
        resultado: INTEGER;
BEGIN
        IF numero < 0 THEN
                RETURN -1;
        END;
        resultado := 1;
        WHILE numero > 1 DO
                resultado := resultado * numero;
                DEC(numero);
        END;
        RETURN resultado;
END Factorial;
```

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

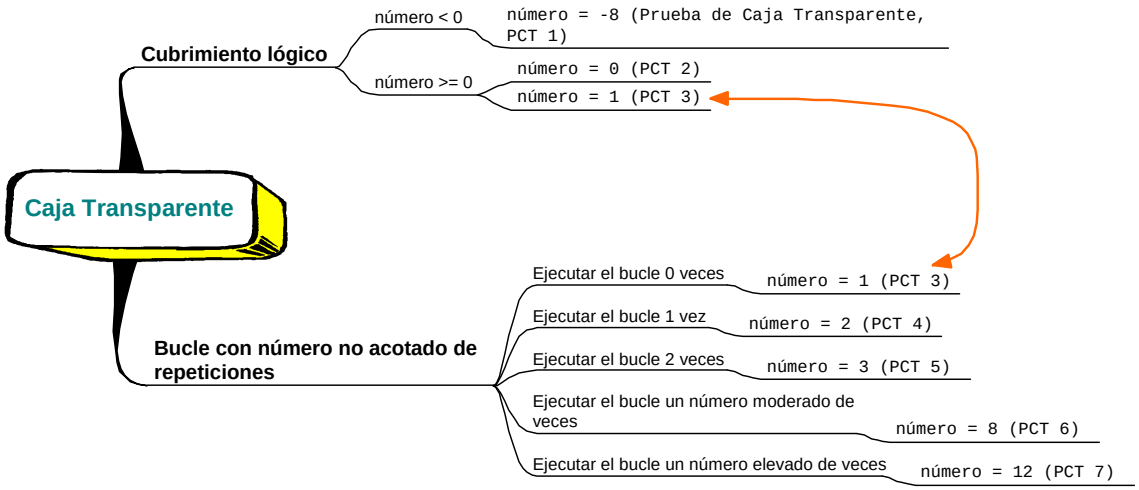
SOLUCIÓN

Pruebas de caja negra:




```
PROCEDURE TestsCajaNegra;
BEGIN
	IF Factorial(-8) <>  -1 THEN
		WriteString("Error en la CE 1");
		WriteLn;
	ELSIF Factorial(0) <>  1 THEN
		WriteString("Error en la CE 2");
		WriteLn;
	ELSIF Factorial(12) <>  479001600 THEN
		WriteString("Error en la CE 3");
		WriteLn;
	END;
END TestsCajaNegra;
```

Pruebas de caja transparente:



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

```
PROCEDURE TestsCajaTransparente;
BEGIN
	IF Factorial(-8) <>  -1 THEN
		WriteString("Error en la PCT 1");
		WriteLn;
	ELSIF Factorial(0) <>  1 THEN
		WriteString("Error en la PCT 2");
		WriteLn;
	ELSIF Factorial(1) <>  1 THEN
		WriteString("Error en la PCT 3");
		WriteLn;
	ELSIF Factorial(2) <>  2 THEN
		WriteString("Error en la PCT 4");
		WriteLn;
	ELSIF Factorial(3) <>  6 THEN
		WriteString("Error en la PCT 5");
		WriteLn;
	ELSIF Factorial(8) <> 40320 THEN
		WriteString("Error en la PCT 6");
		WriteLn;
	ELSIF Factorial(12) <> 479001600 THEN
		WriteString("Error en la PCT 7");
		WriteLn;
	END;
END TestsCajaTransparente;
```

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

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

CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

CÓDIGO CARRERA:
Plan de estudios en extinción: 40=SISTEMAS y 41=GESTIÓN
Plan de estudios NUEVO: 53=SISTEMAS y 54=GESTIÓN

MATERIAL PERMITIDO: NINGUNO

MODELO: EXTRANJERO ORIGINAL
(Londres, Lisboa y Tánger)

Departamento de Ingeniería de Software y Sistemas Informáticos

---

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En 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. La notación de los DFD ¿a qué metodología de análisis está asociada? ¿Cómo se construye un modelo mediante esta notación? ¿Se le ocurre algún procedimiento para identificar los "procesos" y los "flujos de datos" del sistema?

SOLUCIÓN

La notación de los DFD está asociada al análisis estructurado (primer párrafo del epígrafe 4.2.3 "Diseño estructurado", página 165).

Una descripción completa de cómo se construye el modelo, mediante esta notación, aparece en las páginas 52 a 57, epígrafe 2.3.2 "Diagramas de flujo de datos".

Para identificar "procesos" y "flujos de datos" se podría utilizar Abbott (página 173), buscando identificar la información, de distinta naturaleza, que 'fluye' en distintas direcciones (flujos) y las descripciones de las acciones por las cuales se transforman esos flujos de información (los procesos).

2. Defina qué es un Tipo Abstracto de Datos (TAD) y deduzca las relaciones que se puedan establecer con los conceptos de "ocultación", "genericidad", "herencia" y "polimorfismo".

SOLUCIÓN

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.

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

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.

Por tanto, la "ocultación" está garantizada en el TAD, gracias al propio concepto de esta abstracción que oculta la representación de los datos y sólo deja visible las operaciones que manipulan a dichos datos.

Por la propia naturaleza del TAD, se pueden definir tipos de datos que recogen una solución "genérica" o aplicable a varios casos particulares. Un tipo 'genérico' es un tipo parametrizable, es decir, que cubre un abanico de soluciones particulares y se adecua a distintos usos.

En cuanto a la "herencia", un TAD es una abstracción y no está dotada del mecanismo de la herencia.

Por último, la capacidad representar algo o de actuar de distinta forma, sin que se pierda la propia naturaleza (polimorfismo), se puede alcanzar mediante la genericidad. Por tanto, un TAD puede ser polimorfo si es genérico; no es así con respecto al polimorfismo de herencia, del que no dispone un TAD por ser una abstracción.

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

3. Como resultado del diseño arquitectónico de una aplicación hemos caracterizado un módulo llamado 'Gestión de fechas', que realiza las siguientes operaciones sobre una lista de fechas:
   - Buscar la más reciente.
   - Buscar la más antigua.
   - Imprimir todas las fechas que pertenezcan a cierto mes.

Para este módulo, proponga un diseño mediante refinamiento progresivo utilizando:

A. Funciones.

B. Abstracciones.

SOLUCIÓN

A. Tradicionalmente, la atención en el desarrollo del software se centraba en los aspectos funcionales, es decir, se trataba de responder a la cuestión ¿Cómo debe funcionar el programa para comportarse según lo acordado en el análisis? Entre las técnicas o metodologías de diseño en esta época se encuentra la descomposición funcional por refinamientos progresivos.

La descomposición funcional por refinamientos progresivos es una técnica de diseño que proviene de la metodología de programación, desarrollada en los años 60 y 70 principalmente por Dijkstra, y se conoce como programación estructurada. Su principal característica consiste en ver la aplicación compuesta de funciones o procedimientos donde entran unos datos y salen otros transformados. Para lograr esto de forma eficiente y estructurada se diseña atendiendo a las siguientes directrices:

- La especificación de las funciones se expresa mediante un conjunto de construcciones lógicas, las cuales deben garantizar la realización de cualquier tarea y deben ser suficientemente formales para evitar ambigüedades. Para ello se propone como constructores la *secuencia*, la *condición* y la *repetición* empleando para ello un lenguaje en pseudocódigo.
- Se aplica el refinamiento progresivo para el desarrollo de la aplicación, en el siguiente sentido: se plantea inicialmente el programa como una única función global y se va descomponiendo (refinando) poco a poco en funciones más sencillas.

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

- Las estructuras de datos, importantes en cualquier diseño, deben estar supeditadas a la funcionalidad de la aplicación. Estas funciones, utilizan los datos como elementos de entrada para su utilización o trasformación; esto se puede conseguir haciendo que los datos sean variables globales (son vistas desde cualquier función, aunque se pierda cohesión) o se pasan como parámetros de entrada. Estas estructuras de datos se pueden diseñar en cualquier momento y con independencia de la parte operacional.

Primero empezamos planteando el módulo Gestión de Fechas como una función global e iremos descomponiéndola es otras funciones. Para especificar el comportamiento de estas funciones utilizamos pseudocódigo:

```
Gestión de fechas →
     CASO opcion de menu
      SI_ES fecha reciente ENTONCES BuscarFechaReciente
      SI_ES fecha antigua ENTONCES BuscarFechaAntigua
      SI_ES listar fecha del mes ENTONCES ImprimirFechaMes

BuscarFechaReciente →
     FechaInicial = Primer elemento de la Lista de Fechas
       PARA_CADA Elemento EN Lista de Fechas HACER
        SI FechaInicial es MENOR que Elemento ENTONCES
          FechaInicial = Elemento
        FIN-SI
       FIN-PARA

              ...
```

B. Posteriormente, se comenzó a otorgar mayor relevancia a los datos. Es decir, el desarrollo de una aplicación pasó a enfocarse como: ¿qué conceptos o datos intervienen en el programa?, ¿cuáles son sus operaciones o responsabilidades? y ¿cómo se relacionan entre sí?

En un diseño basado en abstracciones, antes de realizar la descomposición funcional u operativa de la aplicación deben quedar identificadas las posibles abstracciones (tipos abstractos de datos, funciones o bien datos encapsulados) que aparecerán en nuestro diseño. Para ello se siguen los siguientes pasos:

i. Identificar claramente las abstracciones que aparezcan en la aplicación. Esto se puede hacer, a partir de las especificaciones, utilizando, por ejemplo, el método de búsqueda de Abbott.
ii. Definir las operaciones asociadas a cada abstracción.
iii. Describir las operaciones mediante lenguaje natural.
iv. A partir de las especificaciones del programa llevar a cabo una descomposición funcional como la descrita en el apartado anterior. Así mismo reemplazar las descripciones de las operaciones de cada abstracción mediante construcciones estructuradas.

Podemos ver el módulo Gestión de fechas como una abstracción funcional global que gestiona a todas las demás abstracciones.

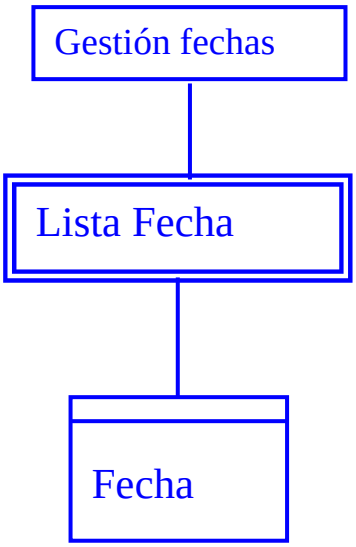
Podemos identificar la lista de fechas como un tipo abstracto de datos, ahora bien, como sólo vamos a tener una lista, podríamos considerarlo un tipo encapsulado. A esta lista estarían asociadas las operaciones de buscar fechas. Otro tipo de datos abstractos sería el que encapsularía tipo fecha y podríamos asociarle, entre otras operaciones, la de comparar con otra fecha.

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

Dato: Lista fecha (Dato encapsulado)
  Atributos
    lista de elementos del tipo fecha
  Operaciones
    Buscar fecha reciente
    Buscar fecha antigua
    Listar fecha mes

Dato: Fecha (Tipo Abstracto de datos)
  Atributos
    Año
    Mes
    Día
  Operaciones
    Comparar con otra fecha

DIAGRAMA DE ABSTRACCIONES



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

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

CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

CÓDIGO CARRERA:
Plan de estudios en extinción: 40=SISTEMAS y 41=GESTIÓN
Plan de estudios NUEVO: 53=SISTEMAS y 54=GESTIÓN

MATERIAL PERMITIDO: NINGUNO

MODELO: AMÉRICA ORIGINAL

Departamento de Ingeniería de Software y Sistemas Informáticos

---

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En 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. Enumere y comente los problemas e inconvenientes que encuentra en el 'ciclo de vida en espiral'.

SOLUCIÓN

La fase de 'Análisis de Riesgos' se ejecuta, normalmente, por una autoridad con experiencia. Esta fase consiste en identificar las fuentes y causas de riesgo, determinar las consecuencias de cada una sobre el desarrollo del producto, establecer un plan de actuación para el caso de que se produzca la situación de riesgo e incorporar la planificación al plan general del desarrollo. Este ciclo de vida incorpora tareas propias de la gestión del desarrollo (planificación y análisis de riesgo) y, por ser mucho más completo y sofisticado que otros modelos, requiere una elaboración más minuciosa. Por otro lado, el análisis de riesgo está estrechamente vinculado a la planificación de la parte que se va a afrontar. Por tanto, la forma de desarrollar el producto se basa, en gran parte, en la propia experiencia del analista de riesgos. Esto puede ser especialmente crítico en las frecuentes ocasiones en las que el analista se ve obligado a 'convencer' al cliente de porqué se planifica y se afronta una parte del desarrollo y no otra.

2. ¿Qué es la validación? ¿Por qué deben incluirse los requisitos de validación en el SRD?

SOLUCIÓN

La 'validación' consiste en la comprobación de que el producto o uno de sus elementos, satisface las especificaciones establecidas por el cliente en la fase de análisis (pág. 16). Normalmente, esas "especificaciones establecidas por el cliente" son las condiciones que él pone para aceptar el producto. Debido al carácter contractual de las mencionadas condiciones de validación, es por lo que resulta muy conveniente que se expresen en documento SRD.

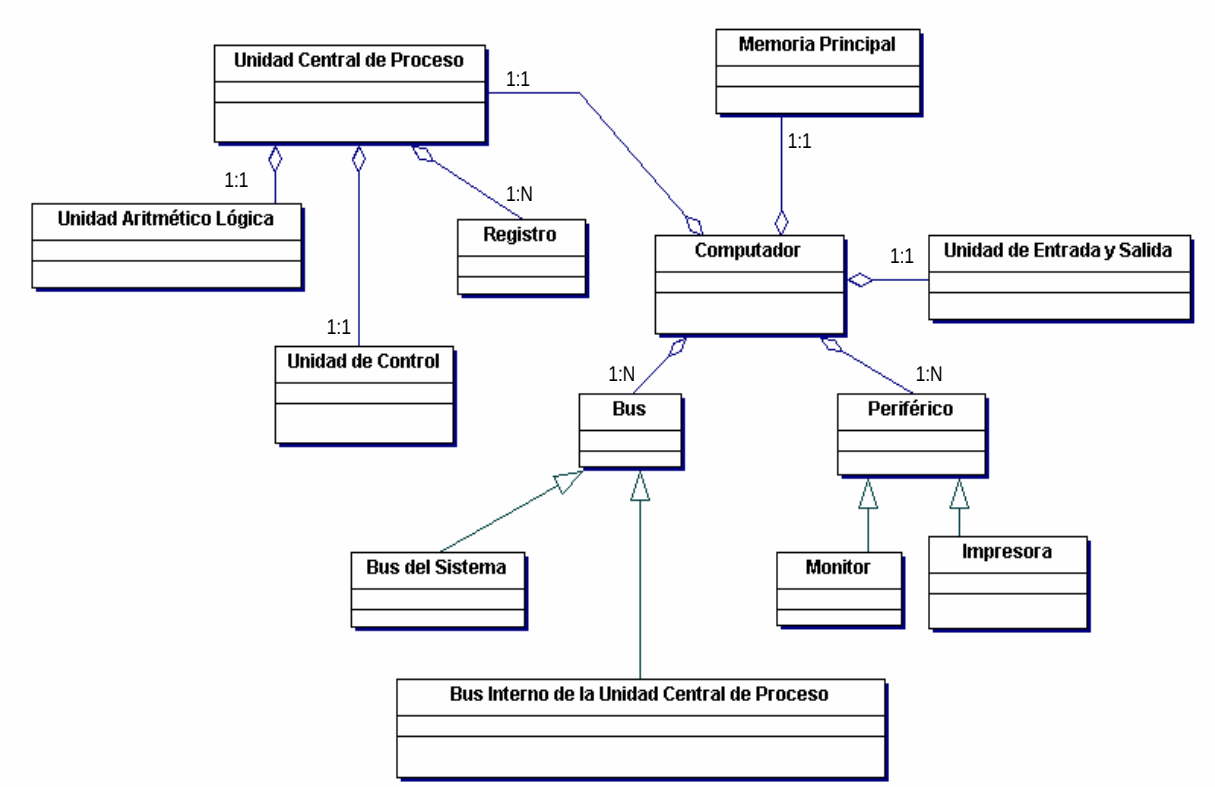
Entregue la hoja de lectura óptica con sus datos junto con su examen.

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

3. Según la arquitectura Von Neumann, un computador está compuesto por:
    1. Unidad Central de Proceso: controla el funcionamiento del computador y lleva a cabo sus funciones de procesamiento de datos. A su vez, consta de:
        a. Unidad Aritmético Lógica: se encarga de realizar operaciones elementales como la suma, resta, AND…
        b. Unidad de Control: se encarga de leer y ejecutar las instrucciones máquina almacenadas en la memoria principal.
        c. Registros: proporcionan almacenamiento interno a la Unidad Central de Proceso.
    2. Memoria Principal.
    3. Unidad de Entrada y Salida: transfiere datos entre el computador y los periféricos.
    4. Periféricos: los hay de distinto tipo, por ejemplo,
        a. Monitor
        b. Impresora
    5. Elementos de interconexión o buses: los hay de distinto tipo, por ejemplo,
        a. Bus del Sistema: conecta Unidad Central de Proceso, Memoria Principal y Unidad de E/S.
        b. Bus Interno de la Unidad Central de Proceso: conecta Unidad Aritmético Lógica, Unidad de Control y Registros.

Desarrolle un modelo de análisis para esta arquitectura Von Neumann. Para ello, utilice la notación de los 'Diagramas de Objetos' que se explica en el texto de la asignatura. No es necesario que incluya los atributos ni las operaciones de las clases. Limítese a representar las relaciones de herencia y composición entre los bloques funcionales descritos anteriormente.

SOLUCIÓN



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

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

CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

CÓDIGO CARRERA:
Plan de estudios en extinción: 40=SISTEMAS y 41=GESTIÓN
Plan de estudios NUEVO: 53=SISTEMAS y 54=GESTIÓN

MATERIAL PERMITIDO: NINGUNO

MODELO: NACIONAL ORIGINAL

Departamento de Ingeniería de Software y Sistemas Informáticos

---

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En cada pregunta teórica, que se valora con 2'5 puntos, 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. Explique qué matices y puntos de vista proyectan sobre el modelo de un mismo sistema: los Diagramas de Flujos de Datos, los Diagramas Entidad - Relación, los Diagramas de Transición de Estados y el Diccionario de Datos.

SOLUCIÓN

Las tres primeras notaciones establecen tres puntos de vista, diferenciados y complementarios, sobre una misma cosa: el modelo del sistema, esto es, la descripción del producto que se va a construir. Como se ve en sinóptico de la izquierda, los diagramas de flujo de datos, aportan el punto de vista de la especificación de los procesos y transformaciones que se producen en la información que fluye. Los diagramas E-R, aportan la descripción de los elementos conceptuales y objetos (entidades) que se manejan en el sistema y las relaciones relevantes que existen entre ellos. Los diagramas de transición de estados, establecen la especificación del control que se establece el sistema; aportan la variable temporal, la evolución de él. Por último, el diccionario de datos complementa y detalla las especificaciones realizadas con cualquiera de las notaciones anteriores, bien sea con lenguajes de descripción funcional o bien con descripciones de datos con cierto formalismo.

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

2. Dadas dos implementaciones alternativas de un algoritmo, ¿coincidirán las pruebas de **caja negra** que verifiquen el funcionamiento correcto de cada implementación? ¿Coincidirán las pruebas si son de **caja transparente**?

SOLUCIÓN

La estrategia de pruebas de caja negra, ignora por completo la estructura interna del programa y se basa, exclusivamente, en la comprobación de la especificación software. En este caso, el **juego de casos de prueba** al que se someten las dos versiones **será el mismo**. En esto consiste el método de 'comparación de versiones' para las pruebas de caja negra. Por el contrario, en la estrategia de caja transparente, los casos de prueba tratan de conseguir que el programa transite por todos los posibles caminos de ejecución y que se pongan en juego todos los elementos del código (que es conocido). Por ello, al depender (los casos de prueba) del código, dichos casos de prueba pueden ser distintos en cada implementación.

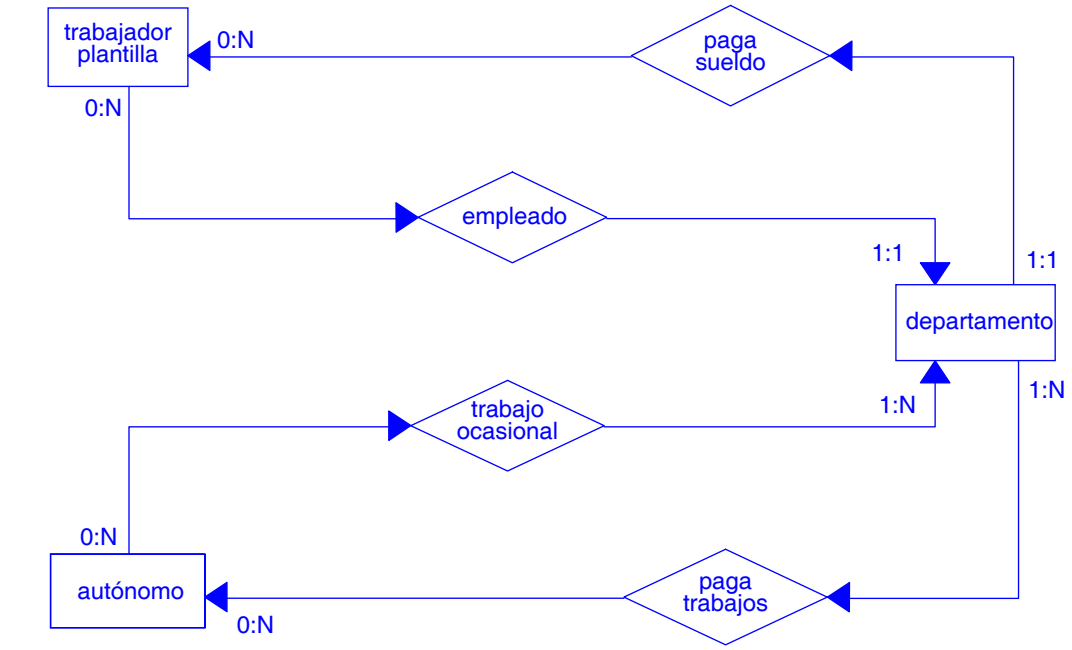
SEGUNDA PARTE. PREGUNTA DE TEORÍA APLICADA (MÁXIMO 5 PUNTOS)

3. Se desea modelar la aplicación de cálculo de nóminas de una empresa según la siguiente descripción:

En la empresa existen trabajadores de plantilla y trabajadores autónomos. Un trabajador de plantilla no puede pertenecer a más de un departamento de la empresa y tiene asignado un sueldo bruto anual. Los autónomos, que realizan trabajos ocasionales para la empresa, pueden hacerlo en distintos departamentos y cobrarán por cada trabajo realizado. Cada departamento dispone de su propio presupuesto para pagar a los trabajadores.

Realice el análisis, mediante diagrama entidad/relación, de la aplicación descrita.

SOLUCIÓN



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

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

CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

CÓDIGO CARRERA:
Plan de estudios en extinción: 40=SISTEMAS y 41=GESTIÓN
Plan de estudios NUEVO: 53=SISTEMAS y 54=GESTIÓN

MATERIAL PERMITIDO: NINGUNO

MODELO: NACIONAL RESERVA

Departamento de Ingeniería de Software y Sistemas Informáticos

---

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En cada pregunta teórica, que se valora con 2'5 puntos, 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. Dé una breve definición de la fase de diseño. Describa cuáles son las cualidades mínimas que se pretende alcanzar con la descomposición modular del diseño.

SOLUCIÓN

El diseño trata de definir y formalizar la estructura de un sistema informático con el suficiente detalle como para permitir su realización física (ultimo párrafo de la página 103, epígrafe 3.1 "Introducción"). Las cualidades mínimas de la descomposición modular son la **independencia funcional**, la **comprensibilidad** y la **adaptabilidad** (epígrafes 4.1.1, 4.1.2 y 4.1.3 en página 150 y siguientes).

2. Enumere las diferencias y coincidencias entre las pruebas α y β

SOLUCIÓN

Coincidencias: son pruebas "de sistema" y se suele aplicar una estrategia de "caja negra". Es el usuario el que realiza las pruebas.

Diferencias: en las pruebas alfa, el entorno de prueba está controlado (se registran las incidencias y se conocen las circunstancias en que se producen) y el usuario recibe el apoyo de alguna persona del equipo de desarrollo, el cual, puede seguir muy de cerca la evolución de las pruebas. En las pruebas beta, el sistema se prueba en el entorno normal de trabajo y sin apoyo de nadie involucrado en el desarrollo. El usuario es el encargado de transmitir, al equipo de desarrollo, las circunstancias en las que se han producido las incidencias (epígrafe 5.9.2 en páginas 290 y 291).

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

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

3. El Sistema de Posicionamiento Global (Global Positioning System, GPS) permite conocer a sus usuarios la posición que ocupan en el globo terráqueo (Figura 1). El sistema consta de, al menos, 24 satélites, 5 estaciones de control terrestres y aparatos receptores. Cada satélite emite periódicamente una señal con información sobre su localización en órbita y el instante en que se ha producido la emisión. Desde la Tierra, un receptor puede conocer su localización si recibe señal desde, al menos, 4 satélites. Conocido el instante en que se emitió una señal y la velocidad a la que viajan las ondas emitidas (la de la luz), el receptor puede calcular su distancia al satélite como:

   distancia = (tiempo de la recepción – tiempo de la emisión) x velocidad de la luz

   Si un receptor recibe señal de un satélite, sabe que se encuentra en la superficie de una esfera de radio la distancia al satélite y centro la localización en órbita del satélite; si recibe señal de 2 satélites, se encuentra en la intersección de dos superficies esféricas, es decir, una circunferencia (Figura 2); si recibe de 3 satélites, las posibilidades se reducen a dos puntos (Figura 3); y, si la recibe de 4, el receptor conoce su posición exacta en el globo terráqueo.

   Normalmente, los receptores guardan un histórico con las localizaciones en órbita de los últimos satélites desde los que recibieron señal. De este modo, cuando se enciende un receptor, éste no inicia una búsqueda desde cero, sino que prueba con los satélites previamente almacenados (sintonizar).

   La órbita que describe un satélite puede sufrir ligeras variaciones. Para mantener la precisión en la localización de los satélites, existen cinco estaciones de control terrestres que ajustan esta información. Concretamente, hay cuatro estacionas automáticas que reciben señales de los satélites y las envían a una estación central que procesa toda la información y transmite las correcciones pertinentes a los satélites.

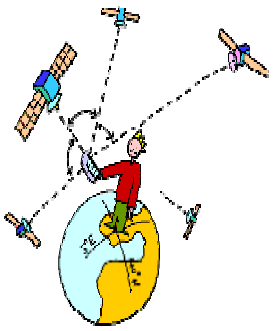


Figura 1

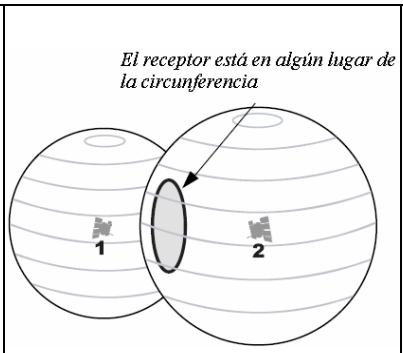


Figura 2

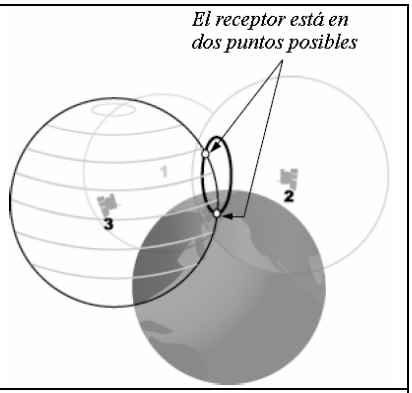


Figura 3

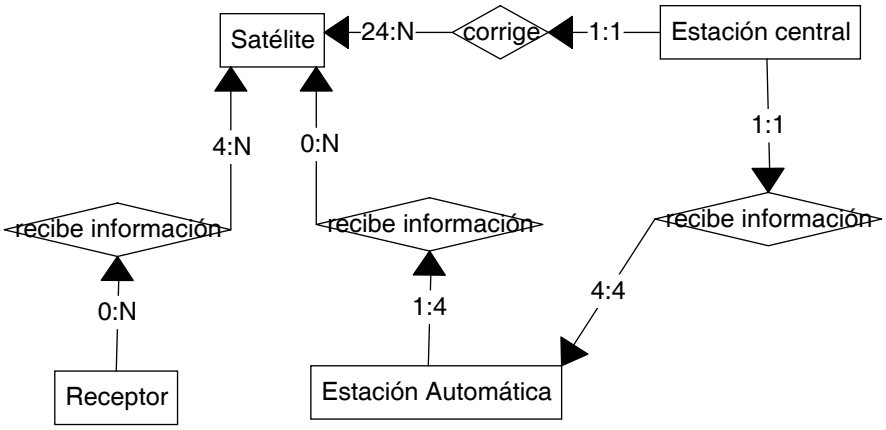
Se pide:

- ***Modelar la descripción anterior mediante un Diagrama Entidad-Relación***
- ***Modelar con un Diagrama de Flujo de Datos el software de un receptor***

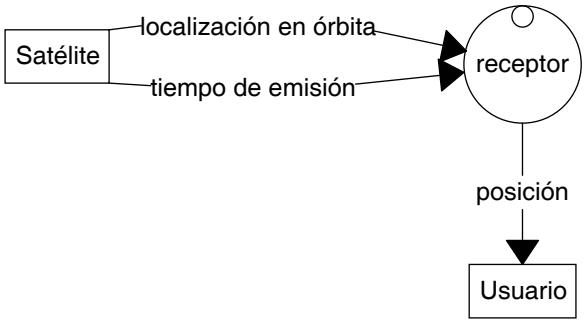
Entregue la hoja de lectura óptica con sus datos junto con su examen.

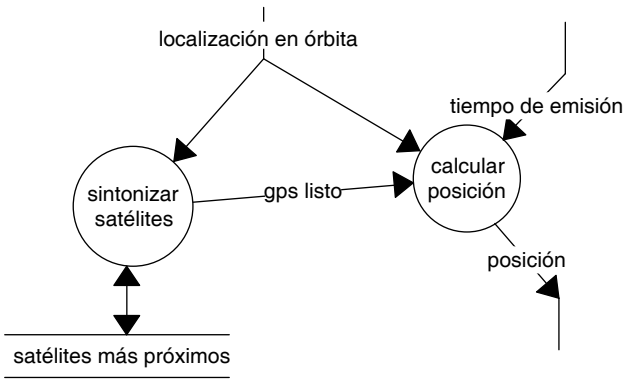
SOLUCIÓN

DER



DFD





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