UF1. [PAC01] SOLUCIÓN

Actividades

Parte teórica

1. Responde a este test en la matriz de respuestas que se encuentra al final de las preguntas, ten en cuenta que puede haber más de una opción correcta:

   1.1. ¿Cuál de las siguientes afirmaciones es correcta?

   a) En los modelos evolutivos no se necesita conocer todos los requisitos al comienzo.
   b) Es muy común en el modelo en cascada el uso de prototipos.
   c) El análisis de riesgos se lleva a cabo en cada incremento del modelo iterativo incremental.
   d) El modelo en cascada es apropiado cuando se necesita una versión inicial del software a desarrollar.

   1.2. ¿Cuál de estas afirmaciones sobre la fase de diseño en el desarrollo de una aplicación es correcta?

   a) En esta fase se especifica qué hay que hacer.
   b) En esta fase se especifica cómo hacerlo.
   c) En esta fase se realiza el proceso de programación.
   d) En esta fase se realizan las pruebas.

   1.3. Cuáles de las siguientes afirmaciones sobre la máquina virtual de Java es cierta:

   a) Un fichero .class contiene código en un lenguaje máquina.
   b) La máquina virtual de Java toma y traduce el bytecode en código binario.
   c) Los ficheros .class sólo pueden ser ejecutados en Microsoft Windows y Linux.
   d) A la hora de instalar el entorno de ejecución de la máquina virtual Java necesitamos saber en qué sistema operativo se va a instalar.

MATRIZ DE RESPUESTAS

1.1	1.2	1.3
A	B	B, D

2. ¿Qué diferencias hay entre el freeware y el software libre?

El software libre es aquel en el que el autor cede una serie de libertades básicas al usuario en el marco de una licencia y establece la libertad para utilizar el programa en varios ordenadores, estudiar cómo funciona y adaptar su código, libertad para distribuir copias a otros usuarios y libertad para mejorar el programa. El software libre deja acceder al código fuente para poder realizar todo lo anterior.

El freeware sin embargo acceder se distribuye sin cargo y normalmente no incluye el código fuente. Suele incluir una licencia de uso que lo restringe. Normalmente no se puede modificar, ni venderla y siempre hay que dar cuenta a su autor. Los programas de software libre no necesariamente son freeware.

3. Realiza un esquema/resumen con los diferentes modelos de ciclo de vida.

Modelo en cascada: las etapas del ciclo de vida siguen un orden, de tal forma que para cambiar de fase se tiene que haber finalizado la anterior.

Modelos evolutivos: se van a ir realizando iteraciones sobre el ciclo de vida del software, permitiendo entregar versiones en cada una de estas iteraciones, mientras se van añadiendo mejoras.

Tipos:

- Modelo iterativo incremental: se compone de varios ciclos de vida en cascada realimentados, en el que cada vez que se realiza el ciclo completo se crea un incremento.
- Modelo en espiral: se combina el modelo en cascada con el modelo iterativo. Cada ciclo de la espiral está formado por 4 fases (determinar objetivos, identificar y resolver riesgos, desarrollar y probar, planificación), y cuando se termina el ciclo, se produce una versión como en el modelo iterativo.

4. Para evitar sorpresas al hacer un proyecto, ¿En qué fases del desarrollo nos tenemos que centrar más? Justifica tu respuesta.

Para evitar sorpresas y obtener un proyecto satisfactorio la fase en la que nos tendremos que centrar más es la fase de Análisis. Es la fase más importante para la obtención de éxito en el proyecto. En ella nos centraremos en entender y comprender el problema que se necesita resolver para después darle una solución. En esta fase se analizan y especifican los requisitos que el programa o sistema deben resolver. La obtención de

estos requisitos no es fácil, pero existen numerosas técnicas como las entrevistas, el desarrollo conjunto de aplicaciones, el brainstorming, etc.

5. ¿El lenguaje Java es un lenguaje compilado o interpretado (u otra opción)? Justifica tu respuesta.

A diferencia de otros lenguajes, Java es compilado e interpretado. Java es compilado cuando su código fuente es traducido a un lenguaje objeto llamado código de máquina (binario, bytecodes) y es interpretado debido a que el código máquina puede ser ejecutado sobre cualquier plataforma la cual debe tener un intérprete ejecutándolo en tiempo real.

6. ¿Qué diferencias hay entre un compilador y un intéprete?

Un compilador es un programa que puede leer un programa escrito en un determinado lenguaje (lenguaje fuente) y traducirlo en un programa equivalente en otro lenguaje (lenguaje destino).

Un intérprete son otra alternativa para traducir los programas escritos en lenguaje de alto nivel, pero en vez de producir un programa destino nos da la apariencia de ejecutar directamente las operaciones especificadas en el programa fuente con las entradas proporcionadas por el usuario.

Los compiladores difieren de los intérpretes en varios aspectos:

- Un programa que ha sido compilado puede correr por si solo, pues en el proceso de compilación se transformó en otro lenguaje (lenguaje máquina).
- Un intérprete traduce el programa cuando lo lee, convirtiendo el código del programa directamente en acciones.
- La ventaja del intérprete es que dado cualquier programa se puede interpretarlo en cualquier plataforma (sistema operativo), en cambio el archivo generado por el compilador solo funciona en la plataforma en donde se lo ha creado.
- Pero por otro lado, un archivo compilado puede ser distribuido fácilmente conociendo la plataforma, mientras que en un archivo interpretado no funciona si no se tiene el intérprete.
- Hablando de la velocidad de ejecución un archivo compilado es de 10 a 20 veces más rápido que un archivo interpretado.

7. ¿De qué tres elementos consta un lenguaje de programación?

Todo lenguaje de programación consta de un alfabeto o vocabulario, también llamado léxico, una sintaxis y una semántica.

- El léxico está formado por el conjunto de símbolos permitidos.
- La sintaxis son las reglas que indican cómo realizar las construcciones con los símbolos del lenguaje.
- La semántica son las reglas que determinan el significado de cualquier construcción del lenguaje.

8. Cita alguna ventaja de usar lenguajes de programación ejecutados por máquinas virtuales.

Los lenguajes de programación ejecutados por máquinas virtuales se pueden ejecutar en cualquier plataforma, es decir, pueden ejecutarse en entornos UNIX, Mac, Windows, etc. Esto es una gran ventaja porque nos permite generar un solo programa que pueda ejecutarse en varios sistemas operativos ya que el encargado de ejecutar el código es la máquina virtual. Un ejemplo sería la JVM (Java Virtual Machine) que es la máquina virtual de Java.

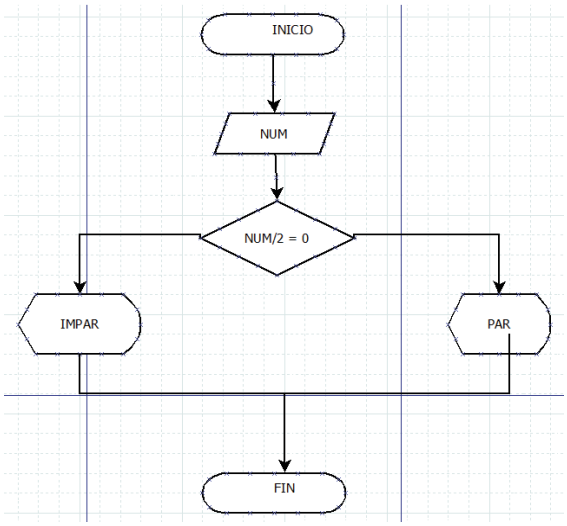
Parte práctica

9. Realiza el pseudocódigo y diagrama de flujo de un programa que diferencie los números pares de los impares.

PSEUDOCÓDIGO

```
Inicio
        Leer NUM
        Si NUM%2 = 0 Entonces
        Visualizar "PAR"
        Si no
        Visualizar "IMPAR"
        Fin si
Fin
```

DIAGRAMA DE FLUJO



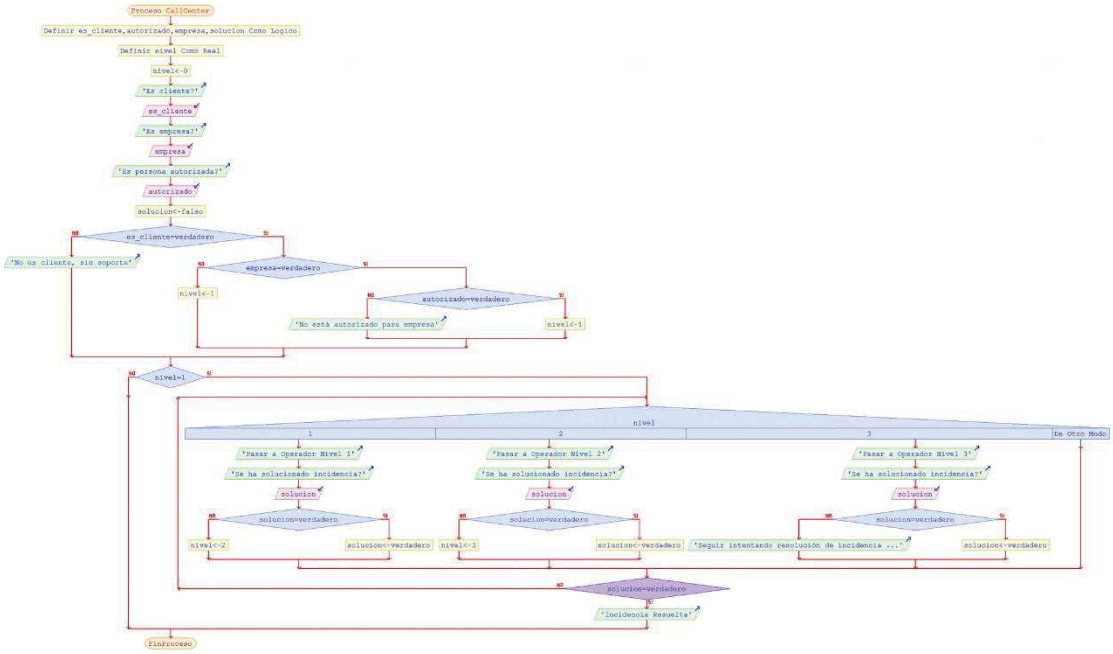
10. Escriba un programa en pseudocódigo y el diagrama de flujo para la gestión de un call center de soporte de una empresa de telefonía. Esta empresa tiene 3 niveles de helpdesk para solucionar las incidencias de los clientes. El operador debe atender la llamada, comprobar que sea cliente de su compañía y que esté autorizado en el caso que el cliente sea una empresa y no el mismo. Si el Operador de nivel 1 no puede resolver el incidente lo escalara a nivel 2 y este a nivel hasta que se resuelva.

PSEUDOCÓDIGO

```
Declarar solución = false
Declarar N_Llamada
Declarar N_Cliente
Visualizar "¿Con quién hablo porfavor?"
Leer N_Llamada
Abrir Fichero Clientes
Leer Registro(Nombre, Empresa, Autorizado)
Si N_Llamada = Nombre
        Si Empresa = false Entonces
        Si no
                Visualizar "¿Su nombre completo?"
                Leer N_Cliente
                Si N_Cliente=Autorizado Entonces
                Si no
                        Visualizar "Sentimos no poder ayudarle"
                Fin si
        Fin si
        Visualizar "N1: ¿Se solucionó el incidente?"
        Leer solución
        Si solución=false Entonces
                Visualizar "Transferimos a N2"
                Visualizar "N2: ¿Se solucionó el incidente?"
```

```
                Leer solución
                Si solución=false Entonces
                        Visualizar “Transferimos a N3”
                        Repetir
                                Visualizar “N3: ¿Solucionó la incidencia?”
                                Leer solución
                        Hasta solución=true
                Fin si
        Fin si
Si no
        Visualizar “Sentimos no poder ayudarle”
Fin si
Visualizar “Gracias por utilizar nuestro servicio”
Cerrar Fichero Clientes
Fin
```

DIAGRAMA DE FLUJO



UF1. [PAC02] SOLUCIÓN

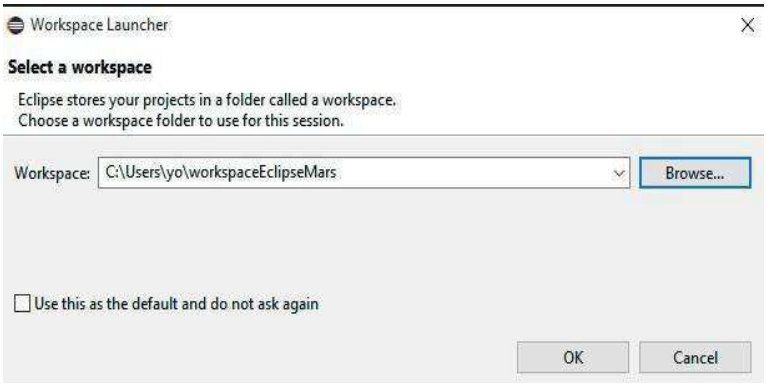
Actividades

Parte práctica

1. Instalación Eclipse.

Aceptamos la licencia.

Una vez instalado, lo lanzamos y seleccionamos la carpeta donde se guardarán los proyectos.



Pantalla de bienvenida. Accedemos a la ventana principal pulsando en Workbench.

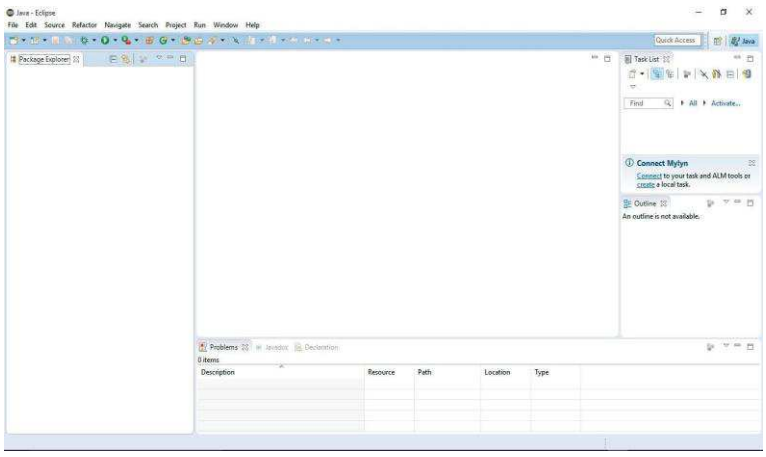
Ventana principal de trabajo.

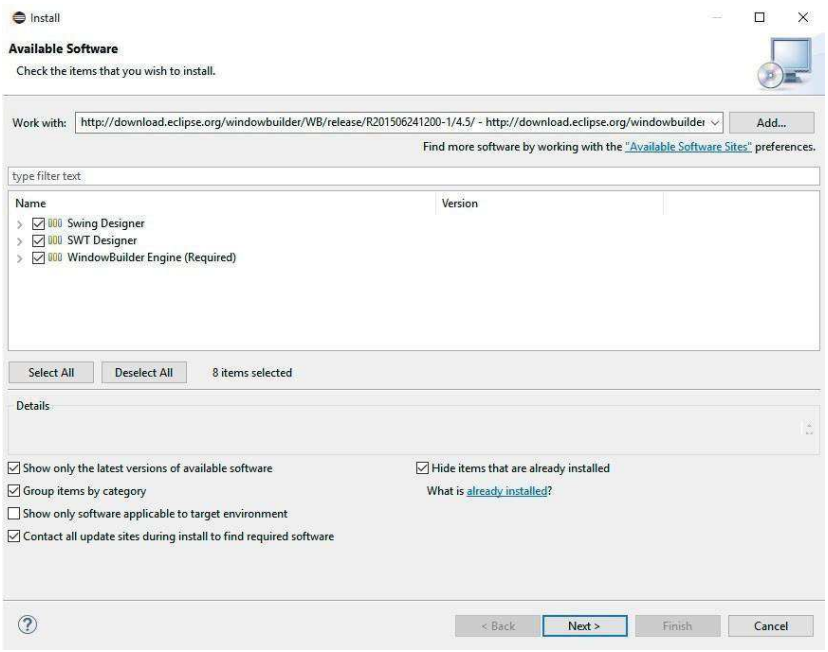
2. Instalación plugins WindowBuilder.

La instalación anterior ya incorpora el plugin WindowBuilder. En caso contrario podemos instalarlo desde Help->Install New Software.

Aparece una ventana donde añadimos el link de descarga http://download.eclipse.org/windowbuilder/WB/release/R201506241200-1/4.5/ en el campo work with, y pulsamos el botón Add.

Después seleccionamos todos los elementos de WindowBuilder y click en Next. Aceptamos los términos de la licencia y click en Finish.



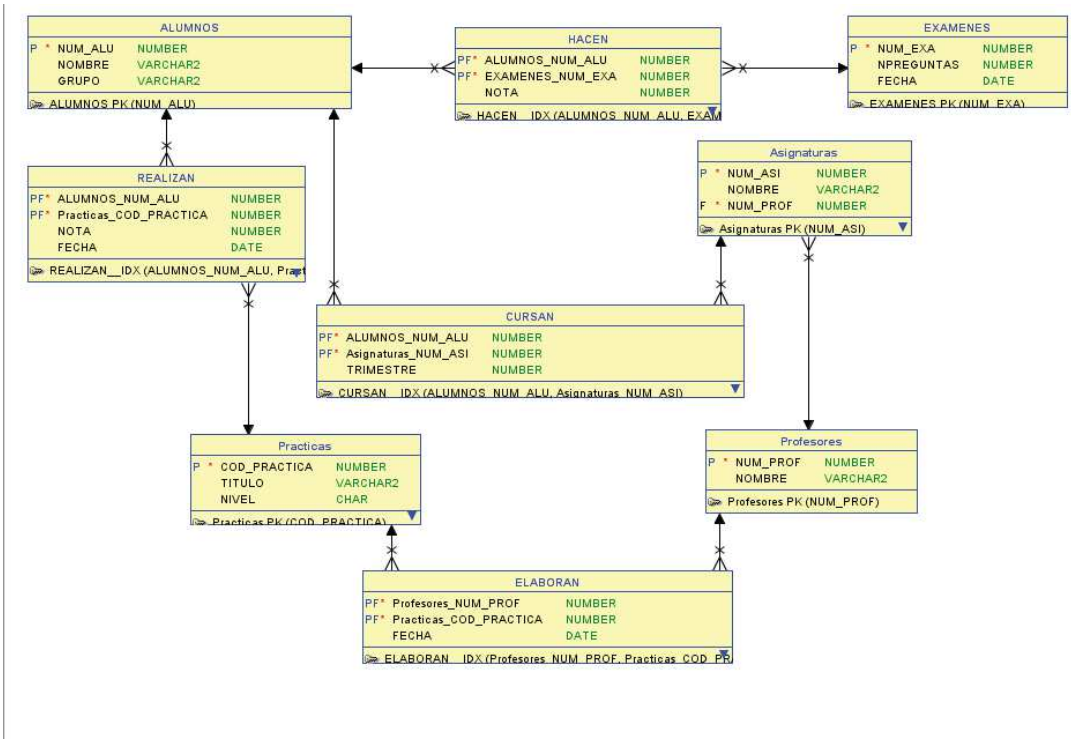


Una vez instalado WindowBuilder creamos un nuevo proyecto java desde **File->New>Java Proyect**.

Después click botón derecho sobre el proyecto y seleccionamos **New->Other.**

En la ventana siguiente: WindowBuilder->Swing Designer->Application Window y next. Finalmente elegimos el nombre de la aplicación, el package y finish.

3. Para el esquema E/R de la página 92 de tu libro de texto a un modelo lógicocon Data Modele, realiza la ingeniería al modelo relacional. Añade los atributos de las relaciones N:M en el modelo relacional (recuerda que las relaciones N:M al pasarlas al modelo relacional se convierten en tablas. Sigue las especificaciones que aparecen en tu libro de texto en la página 92 justo debajo del modelo E/R. Debes entregar el resultado obtenido tras la realización de la ingeniería al modelo relacional.


ALUMNOS
P * NUM_ALU NUMBER
NOMBRE VARCHAR2
GRUPO VARCHAR2
ALUMNOS PK (NUM_ALU)
HACEN
PF* ALUMNOS_NUM_ALU NUMBER
PF* EXAMENES_NUM_EXA NUMBER
NOTA NUMBER
HACEN__IDX (ALUMNOS_NUM_ALU, EXAM
EXAMENES
P * NUM_EXA NUMBER
NPREGUNTAS NUMBER
FECHA DATE
EXAMENES PK (NUM_EXA)
REALIZAN
PF* ALUMNOS_NUM_ALU NUMBER
PF* Practicas_COD_PRACTICA NUMBER
NOTA NUMBER
FECHA DATE
REALIZAN__IDX (ALUMNOS_NUM_ALU, Pra
Asignaturas
P * NUM_ASI NUMBER
NOMBRE VARCHAR2
F * NUM_PROF NUMBER
Asignaturas PK (NUM_ASI)
CURSAN
PF* ALUMNOS_NUM_ALU NUMBER
PF* Asignaturas_NUM_ASI NUMBER
TRIMESTRE NUMBER
CURSAN__IDX (ALUMNOS_NUM_ALU, Asignaturas_NUM_ASI)
Practicas
P * COD_PRACTICA NUMBER
TITULO VARCHAR2
NIVEL CHAR
Practicas PK (COD_PRACTICA)
Profesores
P * NUM_PROF NUMBER
NOMBRE VARCHAR2
Profesores PK (NUM_PROF)
ELABORAN
PF* Profesores_NUM_PROF NUMBER
PF* Practicas_COD_PRACTICA NUMBER
FECHA DATE
ELABORAN__IDX (Profesores_NUM_PROF, Practicas_COD_PR

UF2. [PAC01] Solución

Actividades

Parte teórica

1. Responde a este test en la matriz de respuestas que se encuentra al final de las preguntas:

   1.1. Las pruebas estructurales
      a) Son las pruebas de caja blanca.
      b) Son las pruebas de caja negra.
      c) Son las pruebas de comportamiento.
      d) Sólo miden la entrada y salida.

   1.2. ¿Cómo se llaman las pruebas que se hacen al software en el entorno real de trabajo?
      a) Pruebas unitarias.
      b) Pruebas de integración.
      c) Pruebas de validación.
      d) Pruebas de sistema.

   1.3. ¿Cómo se llaman las pruebas dónde después de probar cada módulo por separado se combinan todos y se prueba el programa completo?
      a) Integración incremental.
      b) Integración no incremental.
      c) Prueba alfa.
      d) Prueba.

   1.4. ¿Cuáles de estos documentos se producen durante el proceso de prueba?
      a) Plan de pruebas.
      b) Especificaciones de prueba.
      c) Informes de pruebas.
      d) Todos los anteriores.

1.5. La prueba de camino básico…

a) Es una técnica de prueba de caja blanca.
b) Es una técnica de prueba de caja negra.
c) Permite obtener una medida de complejidad lógica.
d) Las opciones a y c son correctas.

1.6. El siguiente grafo se corresponde con una estructura

a) HACER MIENTRAS
b) REPETIR HASTA
c) CONDICIONAL
d) SECUENCIAL

MATRIZ DE RESPUESTAS

1.1	1.2	1.3	1.4	1.5	1.6
A	**C**	**B**	**D**	**D**	**A**

2. ¿Qué estrategias se siguen para probar el software? Si las pruebas de unidad funcionan ¿es necesario hacer la prueba de integración?

Las estrategias son: Prueba de Unidad, prueba de Integración, prueba de Validación y prueba del Sistema.

Si es necesario, puesto que es el siguiente paso de las pruebas de software. Se prueban diferentes partes, en las pruebas de Unidad se prueba el código fuente mientras que en las pruebas de Integración se centran en el diseño.

Parte práctica

3. Rellena en la siguiente tabla los casos de prueba tomando como referencia las reglas del análisis de valores límite:

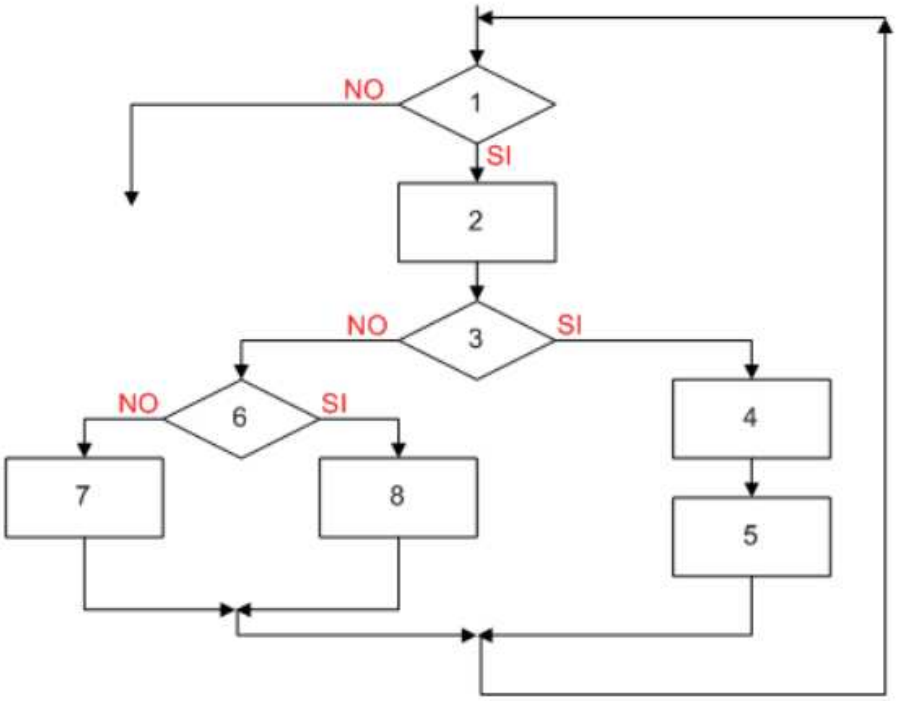
Condiciones de entrada y de salida	Casos de prueba
Una variable toma valores comprendidos entre -4 y 4 (enteros)	Para los valores: -4, 4, -3, 5, -6
El programa lee un fichero que contiene de 1 a 100 registros	Para 0, 1, 10, 100, 101 registros
El programa deberá generar de 1 a 5 listados	Para 0, 1, 5, 6 listados
El número de alumnos para calcular la nota media es 35	Para 0, 1, 35 y 36 alumnos
La función deberá devolver un array de enteros, de 1 a 10 valores	Para 1, 10, 0 y 11 enteros

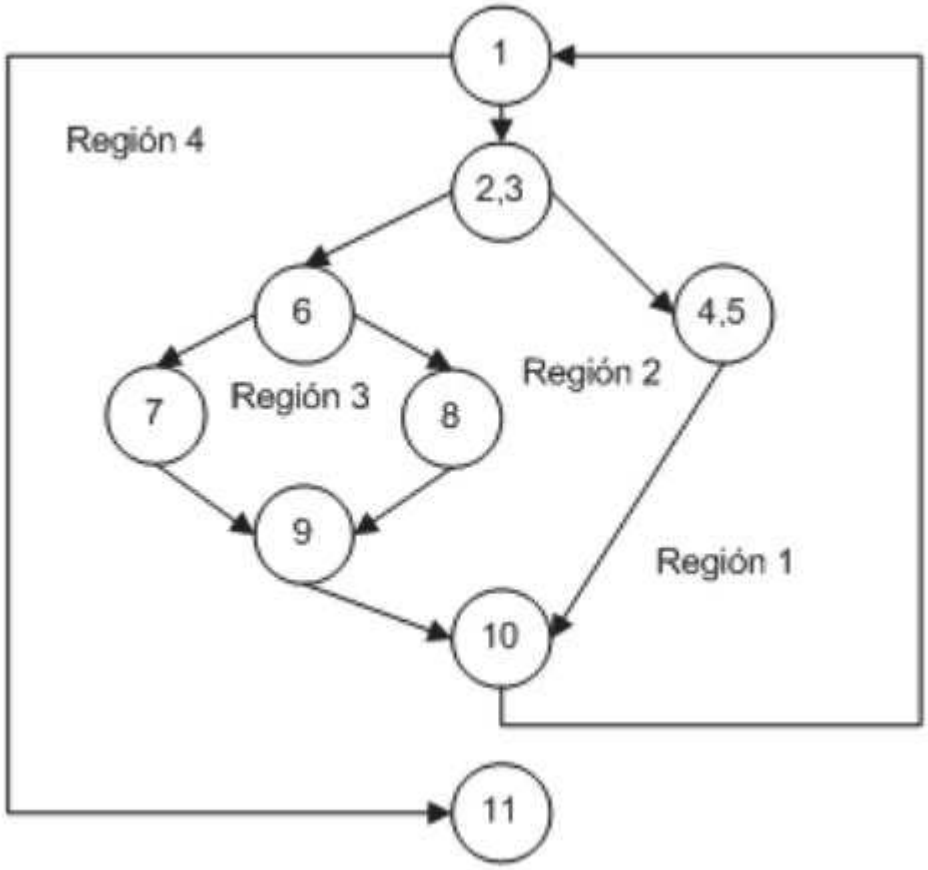
Realiza después un programa Java para probar la función que devuelve el array de enteros. Utiliza los casos de prueba que hayas definido.

```java
public class ArrayEnteros {
	public static void main (String[] args) {
		int n = 10;
		int prueba = -1; //Valores de prueba: -1, 0, 9, 10
		int[] tabla = enteros(n);

		for (int i = 0; i < n; i++) {
			System.out.println(tabla[i]);
		}

		try {
			System.out.println("Elemento: " + tabla[prueba]);

		} catch (NullPointerException e) {
			System.out.println("Array fuera de los límites");
		}
	}

	static int[] enteros (int n) {
		int[] tabla = new int[n];
		int j = 0;

		for (int i = 10; j < n; i++, j++) {
			tabla[j] = i;
		}
```

```
            return tabla;
        }
}
```

4. A partir del siguiente diagrama de flujo, construye el grafo de flujo. Indica el número de nodos, aristas, regiones, nodos predicado, la complejidad ciclomática y el conjunto de caminos independientes.





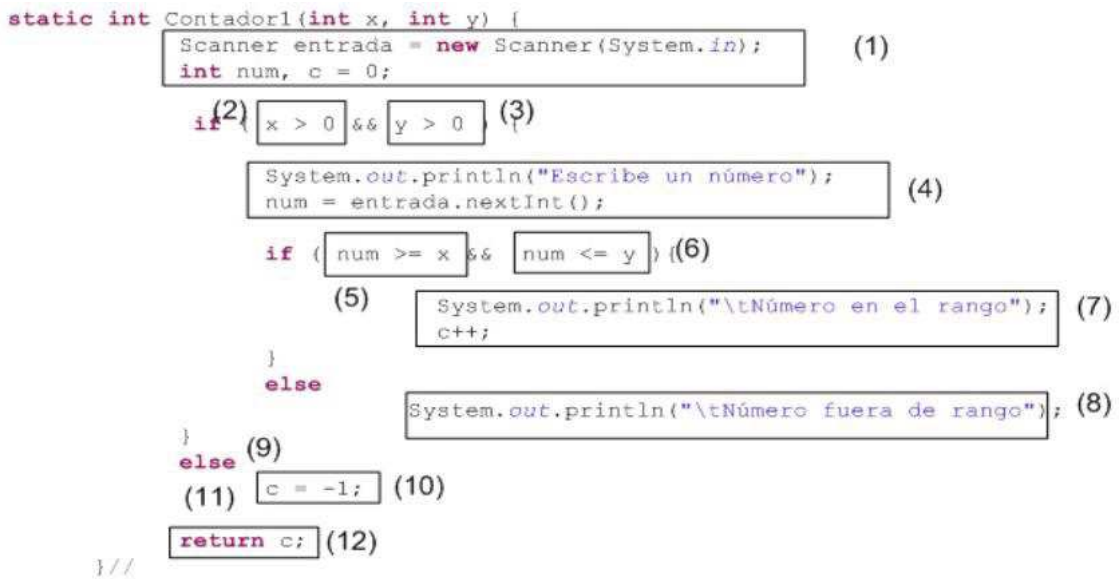
Nº Aristas = 11
Nº Nodos = 9
Nodos predicado = 3 → Nodo que contiene una condición (1; 2,3; 6)
Nº regiones = 4

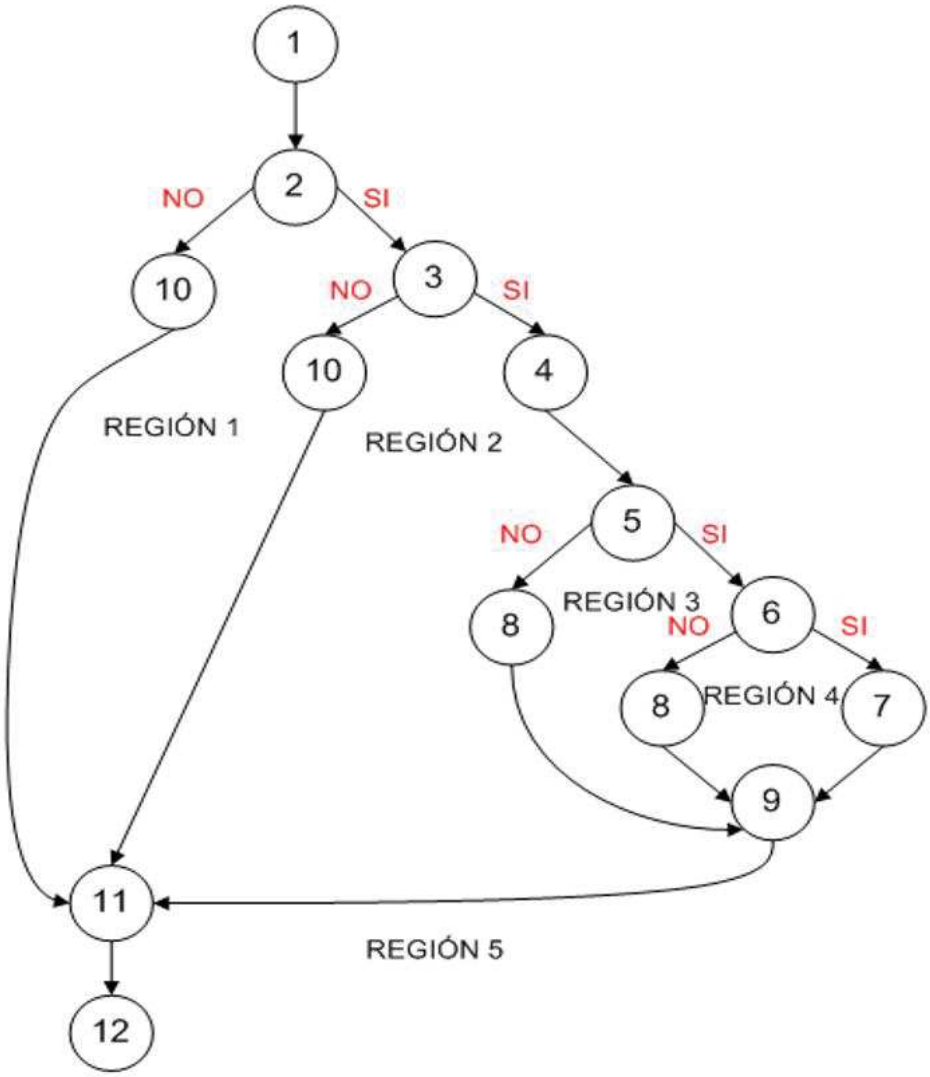
V(G) = 4, tenemos 4 caminos básicos, sin mucho riesgo.

Camino 1: 1 - 11
Camino 2: 1 - 2,3 - 4,5 - 10 - 1 - 11
Camino 3: 1 - 2,3 - 6 - 8 - 9 - 10 - 1 - 11
Camino 4: 1 - 2,3 - 6 - 7 - 9 - 10 - 1 - 11

5. Realiza el grafo de flujo, calcula la complejidad ciclomática, define el conjunto básico de caminos, elabora los casos de prueba para cada camino y evalúa el riesgo para la siguiente función Java:

```
static int Contador1(int x, int y) {
  Scanner entrada = new Scanner(System.in);
  int num, c = 0;
  if (x > 0 && y > 0) {
      System.out.println("Escribe un número");
      num = entrada.nextInt();
      if (num >= x && num <= y){
           System.out.println("\tNúmero en el rango");
           c++;
      }
      else
           System.out.println("\tNúmero fuera de rango");

      }
  else
      c = -1;
  return c;
}//
```





Nº Aristas = 17
Nº Nodos = 12
Nodos predicado = 4 → Nodo que contiene una condición (2; 3; 5; 6)
Nº Regiones = 5
V(G) = 5, tenemos 5 caminos básicos, sin mucho riesgo

Camino 1: 1 - 2 - 10 - 11 - 12
Camino 2: 1 - 2 - 3 - 10 - 11 - 12
Camino 3: 1 - 2 - 3 - 4 - 5 - 8 - 9 - 11 - 12
Camino 4: 1 - 2 - 3 - 4 - 5 - 6 - 8 - 9 - 11 - 12
Camino 5: 1 - 2 - 3 - 4 - 5 - 6 - 7 - 9 - 11 - 12

Camino	Caso de prueba	Resultado esperado
Camino 1	**Escoger algún X tal que NO se cumple la condición X>0** **X = -1** **Contador (-1, 10)**	**C = -1** **Devuelve -1**
Camino 2	**Escoger algún X e Y tal que se cumpla la condición X > 0 y NO se cumpla Y > 0** **X = 1, Y=-10** **Contador (1, 10)**	**C = -1** **Devuelve -1**
Camino 3	**Escoger algún X e Y tal que se cumpla la condición X > 0 e Y > 0** **X = 1, Y = 10, num 5** **Contador (1,10)**	**Número en el rango** **C = 1** **Devuelve 1**
Camino 4	**Escoger algún X e Y tal que se cumpla la condición X > 0 e Y > 0** **X = 1, Y = 10, num = 15** **Contador (1,10)**	**Número en el rango** **C = 0** **Devuelve 0**
Camino 5	**Escoger algún X e Y tal que se cumpla la condición X > 0 e Y > 0** **X = 1, Y = 10, num = -15** **Contador (1,10)**	**Número fuera de rango** **C = 0** **Devuelve 0**

UF2. [PAC01] Solución

Actividades

Parte teórica

1. ¿Qué es el control de versiones?

La capacidad de recordar los cambios realizados tanto en la estructura de directorios como en el contenido de los archivos.

2. ¿Qué es subversión? ¿Cómo funciona el ciclo de vida en subversión?

Es una herramienta multiplataforma de código abierto para el control de versiones.

El ciclo de vida de una subversión parte del comienzo del software, se crea el tronco del proyecto (trunk) y después se van añadiendo funcionalidades a este desarrollo mediante las ramas, para que no dañen el proyecto. Cada vez que se tienen listas (sin bugs) estas funcionalidades se fusionan con la rama principal. Por último, cada vez que hay una nueva versión del software, se añade una etiqueta para poder identificar estas partes fácilmente.

3. En control de versiones qué significa:

3.1. Repositorio.

Lugar donde se almacenan todos los datos y los cambios.

3.2. Revisión, versión.

Es una versión concreta de los datos almacenados.

3.3. Checkout.

Crear una copia de trabajo del proyecto, o de archivos y carpetas del repositorio en el equipo local.

3.4. Commit.

Se realiza commit cuando se confirman los cambios realizados en local para integrarlos al repositorio.

3.5. Import.

Es la subida carpetas y archivos del equipo local al repositorio.

3.6. Actualizar.

Se realiza una actualización cuando se desea integrar los cambios realizados en el repositorio en la copia de trabajo local.

3.7. Crear una rama.

Son copias del proyecto, por lo que se crea una bifurcación entre el proyecto y su copia, y se puede modificar cualquiera de las dos vertientes.

3.8. Crear una etiqueta.

Identificar puntos del proyecto fácilmente.

3.9. Crear un conflicto.

Dos usuarios trabajan sobre la misma copia en local, sin que uno de ellos actualice su versión cuando el otro usuario lo ha modificado en el repositorio.

4. ¿Qué es documentar el código de un programa?

Explicar claramente lo que hace el programa, de esta manera todo equipo de desarrollo sabrá lo que se está haciendo y por qué.

5. ¿Qué tipos de documentación se realizan en los proyectos?
   - Documentación de las especificaciones.
   - Documentación del diseño.
   - Documentación del código fuente.
   - Documentación de usuario final.

6. ¿Qué es JavaDoc? ¿Para qué se utiliza?

Es una utilidad de Java que se utiliza para extraer y generar documentación directamente del código en formato HTML.

7. ¿Qué es refactorización?

Es una técnica de la ingeniería de software que permite la optimización de un código previamente escrito.

- Limpia el código, mejorando la consistencia y la claridad.
- Mantiene el código, no corrige errores ni añade funciones nuevas.
- Va a permitir facilitar la realización de cambios en el código.
- Se obtiene un código limpio y altamente modularizado.

8. ¿Cuándo hay que refactorizar?

- Cuando detectamos el mismo código en distintos lugares.
- Cuando los métodos son muy largos.
- Cuando una clase tiene muchos métodos, atributos e instancias.
- Cuando se pasan muchos parámetros.
- Cuando una clase es frecuentemente modificada.
- Cuando se deben realizar cambios para compatibilizar con otros cambios en otras clases.
- Cuando un método utiliza más elementos de otra clase que de la suya propia.
- Cuando tenemos clases que sólo tienen atributos y métodos de acceso a ellos.
- Cuando una subclase utiliza pocas características de sus superclases.

9. ¿Qué son los bad smells?

Son los síntomas para refactorizar:

- Código duplicado.
- Métodos muy largos.
- Clases muy grandes.
- Lista de parámetros extensa.
- Cambio divergente.
- Cirugía a tiro pistola.
- Clase de solo datos.
- Legado rechazado.

10. Cita métodos de refactorización en eclipse.

- Rename
- Move
- Extract Constant
- Extract Local Variable
- Convert Local Variable to Field
- Extract Method
- Change Methos Signature
- Inline
- Member Type to Top Level
- Extract Interface
- Extract Superclass
- Convert Anonymous Class to Nested

UF3. [PAC01] SOLUCIÓN

Actividades

Parte teórica

1. Responde a este test en la matriz de respuestas que se encuentra al final de las preguntas:

   1.1. ¿En qué diagrama podemos observar cómo se relacionan las clases entre sí?

   a) Diagramas de clase.
   b) Diagramas de caso de uso.
   c) Diagramas de despliegue.
   d) Diagramas de paquetes.

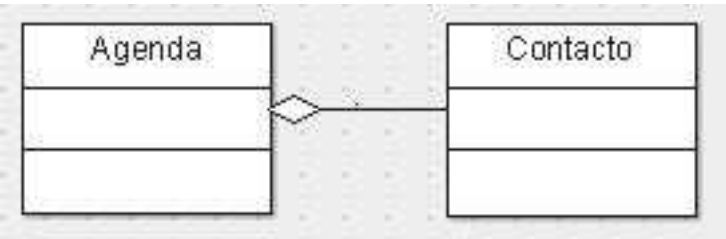
   1.2. ¿En qué tipo de diagrama podemos mostrar la secuencia de actividades?

   a) Diagramas de objetos.
   b) Diagramas de secuencia.
   c) Diagramas de estado.
   d) Diagramas de actividad.

   1.3. ¿Qué es una clase asociación?

   a) Una asociación con relación 1:1..*
   b) Una clase con información necesaria para una asociación entre otras clases.
   c) Una clase que se asocia consigo misma.
   d) Una clase con relación 0..1:N

1.4. Si tenemos la siguiente relación:



a) La clase contacto hereda de la clase agenda.
b) Una agenda se compone de contactos.
c) Las agendas se componen de contactos.
d) Todas las anteriores.

1.5. ¿Cuáles son los estereotipos de los diagramas de comportamiento?

a) Enumeration e interface.
b) Entity, control y boundary.
c) Entity, control y enumeration.
d) Entity, interface y boundary.

1.6. ¿Con cuál de estos programas no podemos realizar diagramas de clases?

a) ArgoUML
b) Eclipse
c) WhiteStarUML
d) Con todos los programas anteriores es posible realizar diagramas de clases.

1.7. La flecha de dependencia

a) Va desde la clase utilizada a la clase que la utiliza.
b) Va desde la clase que utiliza a la clase utilizada.
c) Se representa con una flecha sin relleno.
d) Las opciones b y c son correctas.

1.8. ¿Qué significa un # delante de un atributo?

a) Es un atributo con visibilidad public.
b) Es un atributo con visibilidad private.
c) Es un atributo con visibilidad protected.
d) Es un atributo con visibilidad de paquete.

1.9. Las relaciones que tenemos entre clases pueden ser

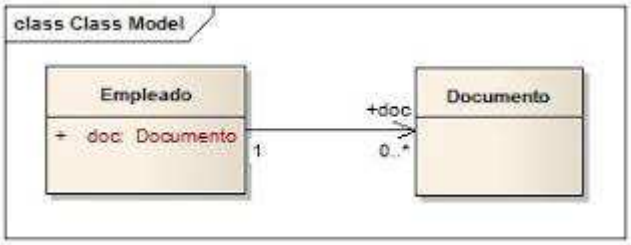
a) Asociación y realización.

b) Herencia y dependencia.

c) Agregación y composición.

d) Todas son correctas.

1.10. Selecciona la respuesta verdadera:



a) Empleado a Documento es navegable y Documento a Empleado es navegable.

b) Empleado a Documento es navegable, pero Documento a Empleado no es navegable.

c) Empleado a Documento no es navegable, pero Documento a Empleado sí.

d) Empleado a Documento no es navegable, al igual que Documento a Empleado.

MATRIZ DE RESPUESTAS

1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	1.10
A	**D**	**B**	**C**	**B**	**D**	**D**	**C**	**D**	**B**

Parte práctica

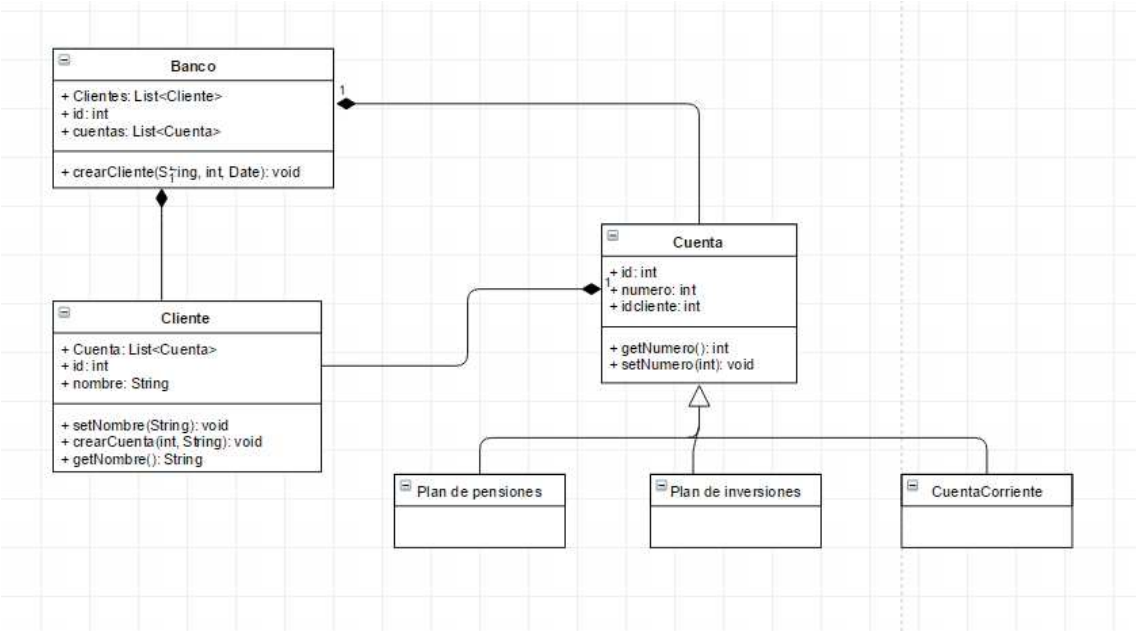
2. Realice un diagrama de clases para un banco:

   a. El banco tendrá varios clientes.

   b. El cliente puede tener varias cuentas.

   c. El banco, además de tener cuentas corrientes, pueden establecer planes de pensiones y planes de inversiones.

Nota: Cada alumno establecerá los atributos y métodos que crea necesarios para cada clase.



UF3. [PAC02] Solución

Actividades

Parte teórica

1. Responde a este test en la matriz de respuestas que se encuentra al final de las preguntas:

   1.1. ¿Cuál de las siguientes afirmaciones sobre los diagramas de estado es correcta?

   a) Son adecuados para describir el comportamiento de varios objetos en un mismo caso de uso.
   b) Son adecuados para describir el comportamiento de un objeto.
   c) Son adecuados para mostrar la secuencia general de acciones de varios objetos y casos de uso.
   d) Todas son correctas.

   1.2. ¿Cuál de las siguientes afirmaciones sobre los diagramas de interacción es correcta?

   a) Son adecuados para describir el comportamiento de varios objetos en un mismo caso de uso.
   b) Son adecuados para describir el comportamiento de un objeto.
   c) Son adecuados para mostrar la secuencia general de acciones de varios objetos y casos de uso.
   d) Todas son correctas.

   1.3. ¿Cuál de las siguientes afirmaciones sobre los diagramas de actividad es correcta?

   a) Son adecuados para describir el comportamiento de varios objetos en un mismo caso de uso.
   b) Son adecuados para describir el comportamiento de un objeto.
   c) Son adecuados para mostrar la secuencia general de acciones de varios objetos y casos de uso.
   d) Todas son correctas.

MATRIZ DE RESPUESTAS

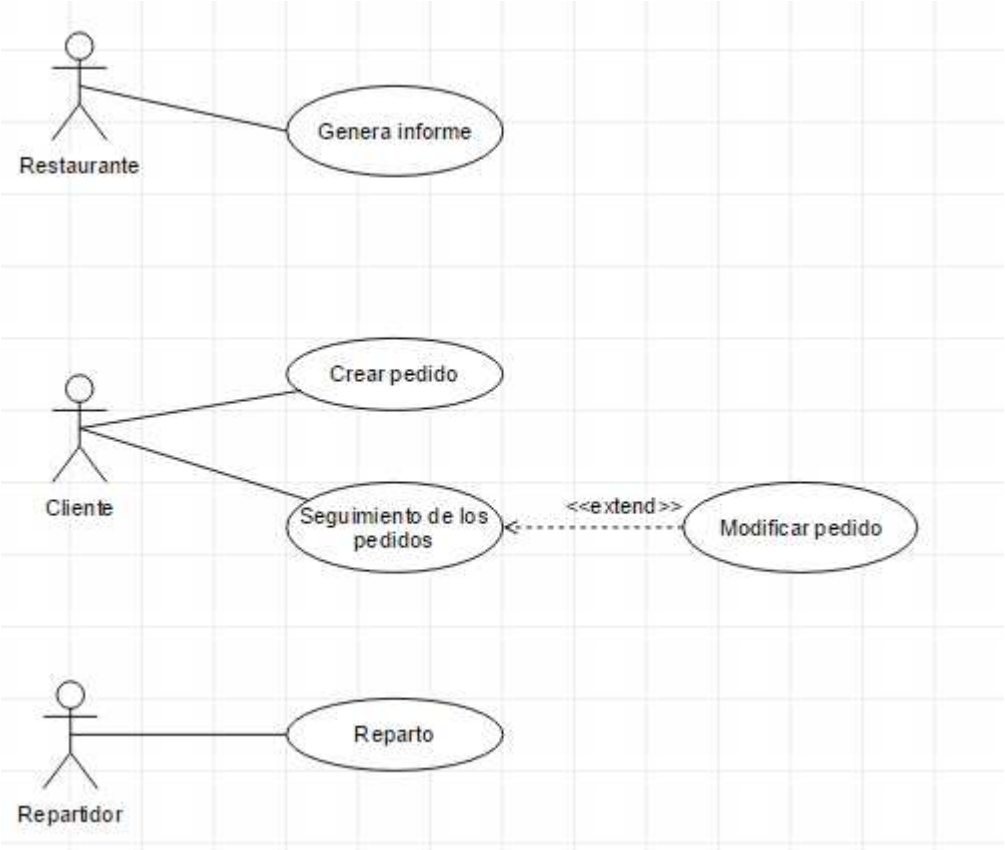
1.1	1.2	1.3
B	A	C

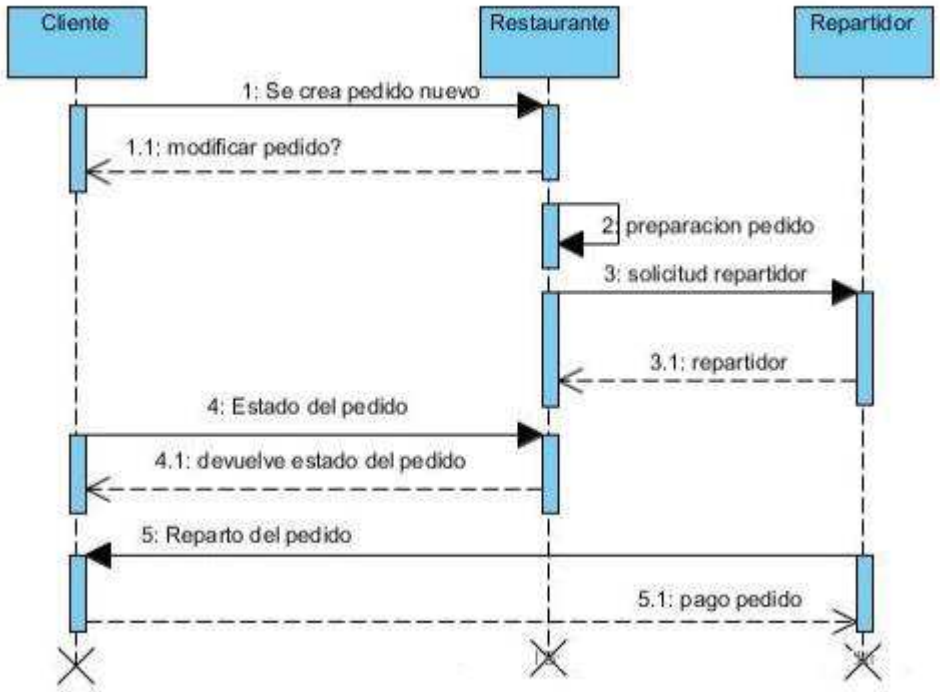
Parte práctica

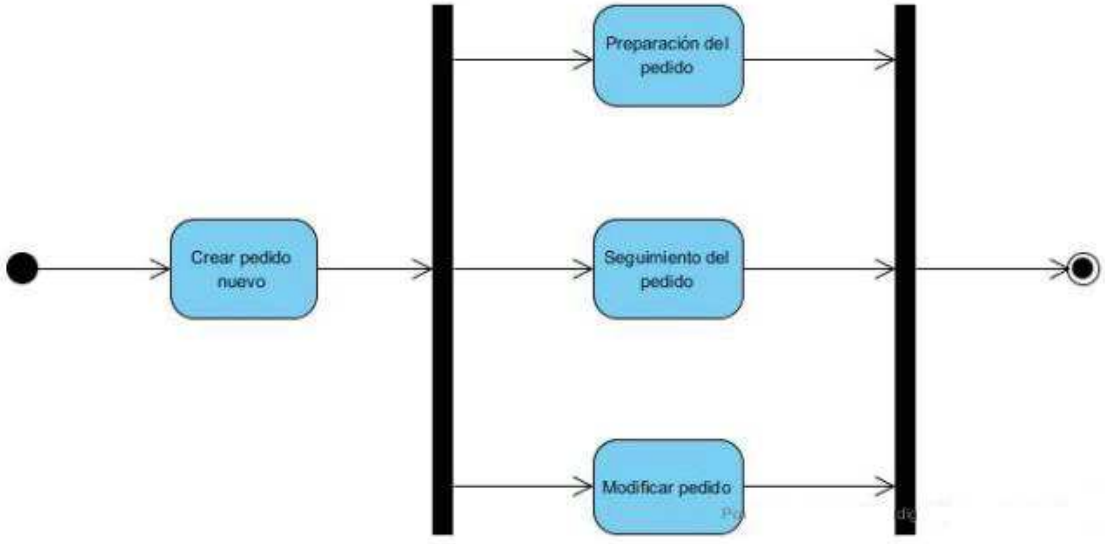
2. Necesitamos una aplicación que permite a un restaurante gestionar los pedidos creados por sus clientes permitiéndoles llevar un seguimiento de su pedido y modificar los pedidos antes de que se repartan. Además, quiere llevar un seguimiento de los pedidos repartidos por sus empleados mediante informes que generará la propia aplicación. Realice:

   2.1. El diagrama de casos de uso y de secuencia para el restaurante.

   2.2. El diagrama de actividad para sus pedidos.




Cliente
Restaurante
Repartidor
1: Se crea pedido nuevo
1.1: modificar pedido?
2: preparacion pedido
3: solicitud repartidor
3.1: repartidor
4: Estado del pedido
4.1: devuelve estado del pedido
5: Reparto del pedido
5.1: pago pedido


Crear pedido nuevo
Preparación del pedido
Seguimiento del pedido
Modificar pedido

UF3. [PAC03] Solución

Actividades

Parte práctica

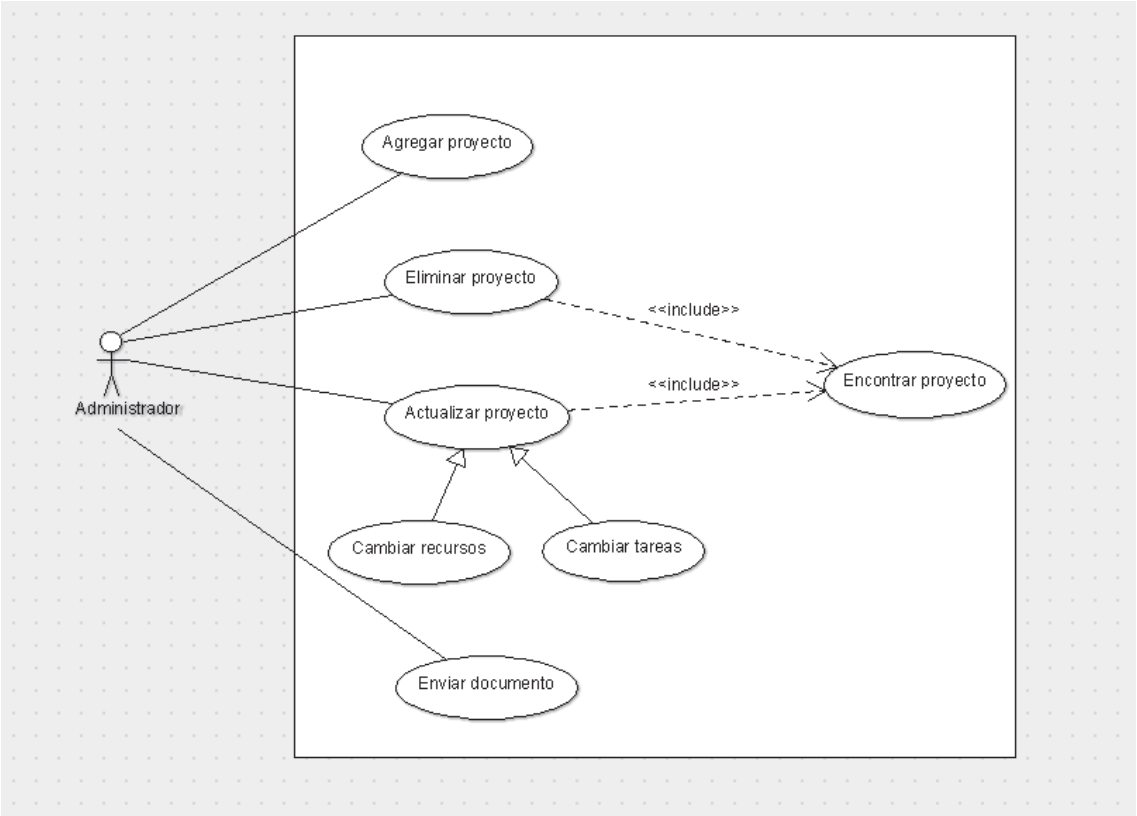
1. Construye el diagrama de casos de uso para el siguiente enunciado: en una oficina se lleva a cabo la gestión de proyectos. La única persona que controla los proyectos es el administrador, cuyas funciones son las siguientes:

    - Puede agregar, eliminar y actualizar un proyecto, pero para eliminar y actualizar es necesario encontrar el proyecto en cuestión.

    - A la hora de actualizar un proyecto se pueden dar dos situaciones: cambiar la información sobre las tareas del proyecto o cambiar los recursos asociados al proyecto.

    - Para informar a todos los miembros del equipo sobre los avances en el proyecto se envía un documento por e-mail.

    Utiliza las relaciones <<include>>, <<extend>> y de generalización que consideres.



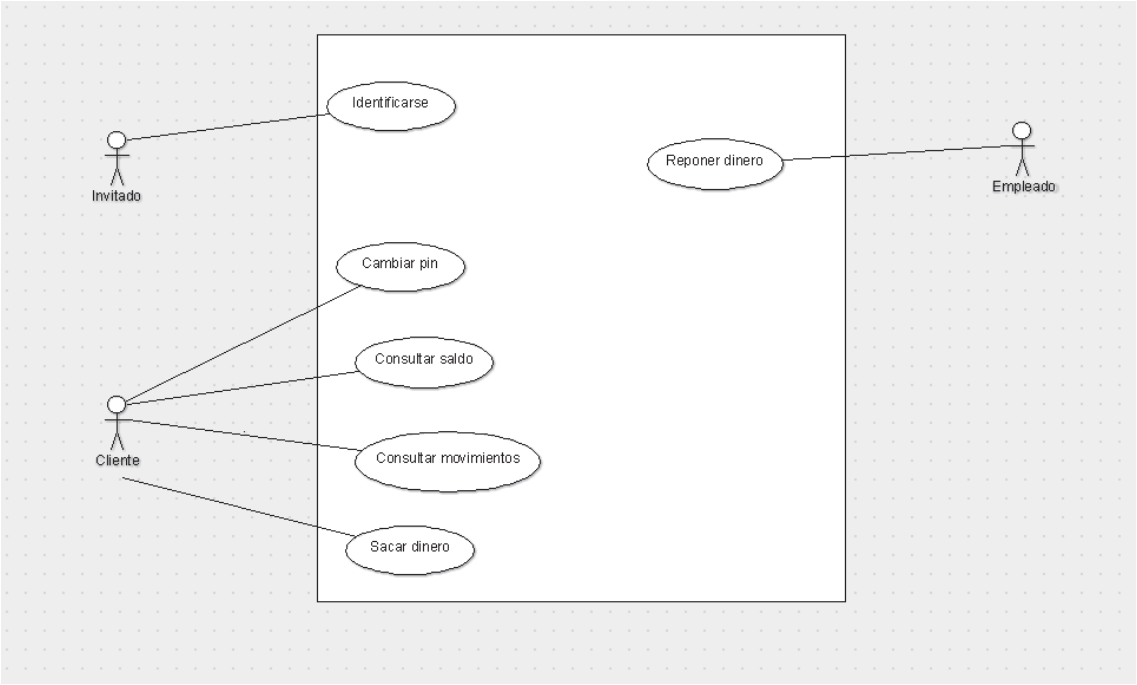
2. Escribe un diagrama de casos de uso para modelar la interacción de un cliente y un empleado de un banco con un cajero automático. Las especificaciones son las siguientes:

    - El cajero automático lo puede utilizar el cliente del banco y el empleado de la sucursal.

    - El cliente debe identificarse en el cajero antes de realizar cualquier operación.

    - El cliente puede cambiar el pin, sacar dinero, consultar el saldo y consultar los últimos movimientos.

    - El empleado utiliza el cajero automático únicamente para reponer dinero.


Identificarse
Invitado
Reponer dinero
Empleado
Cambiar pin
Consultar saldo
Cliente
Consultar movimientos
Sacar dinero