Universidad de los Andes

Ingeniería de Sistemas y Computación

Algorítmica y Programación por Objetos 2

Hoja de Trabajo Nivel 1: El parqueadero

Proyecto Cupi2

Integrantes:

El objetivo principal de esta guía de trabajo es repasar los conceptos vistos en el primer curso de programación. Con este trabajo el estudiante podrá:

- Enumerar los conceptos básicos en la solución de problemas con programas de computador
- Describir el proceso de desarrollo seguido en el curso
- Especificar apropiadamente un problema que se quiere solucionar
- Modelar el mundo de un problema usando una aproximación de objetos
- Implementar una solución a un problema en lenguaje Java
- Implementar una interfaz gráfica en Java
- Utilizar Eclipse como ambiente de desarrollo

La guía se divide en dos partes: La primera es un trabajo en clase guiado por este documento, y la segunda parte para desarrollar en los laboratorios con el objetivo de repasar el trabajo con la herramienta Eclipse.

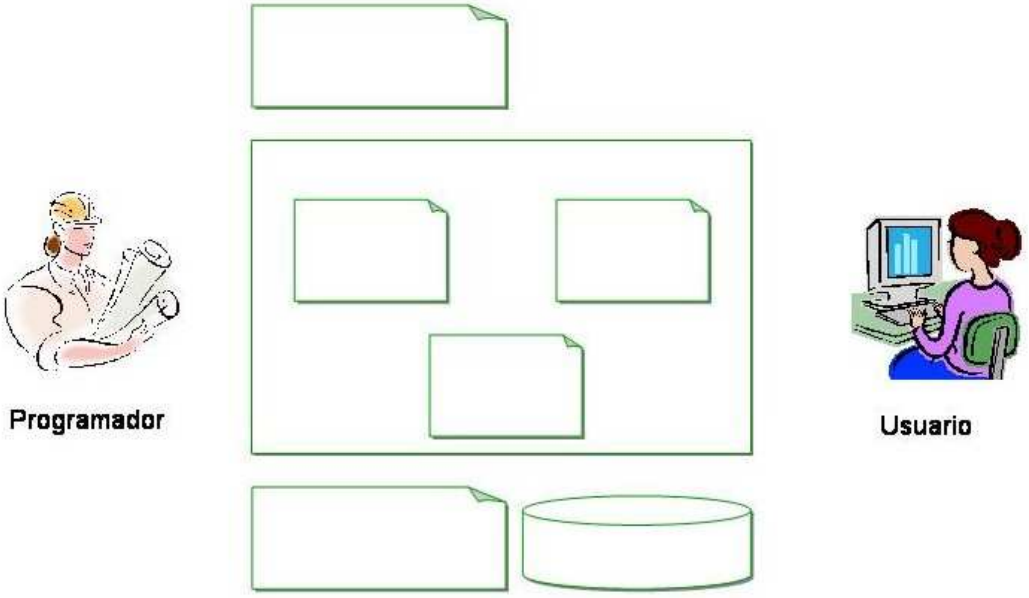
Construir mapa conceptual sobre programación	*Identifique relaciones entre los siguientes conceptos, conectándolos con flechas nombradas*

Describir el proceso de solucionar un problema	*Describa las etapas requeridas para la construcción de una solución computacional a un problema y los productos de cada una de ella*

¿Cuál es la herramienta que nos permite escribir y probar los programas?

¿Qué otras herramientas podrían ser útiles para el proceso?

Describir los elementos que conforman la solución al problema	*Complete el siguiente esquema para describir los elementos de la solución*



Especificar el problema	*Lea el siguiente enunciado, que describe el problema a resolver y resuelva los puntos a continuación*

Se quiere construir una aplicación para administrar un parqueadero. Dicho parqueadero tiene 87 puestos numerados del 1 al 87. En cada puesto se puede parquear un sólo carro. El parqueadero tiene una tarifa por hora o fracción de hora, que cambia cada vez que el gobierno lo autoriza.
Cada carro se identifica por su placa, pero también se debe conocer la hora en la que entró, que corresponde a un valor entre 6 y 20, dado que el parqueadero está abierto entre 6 de la mañana y 8 de la noche.
Se espera que con la aplicación que se quiere construir se pueda hacer lo siguiente: (1) A un carro que llega, decirle el puesto en el que se debe parquear (si hay cupo), (2) A un carro que sale, decirle cuánto debe pagar, (3) Al administrador del parqueadero, decirle cuanta plata se ha recogido en el día, (4) Al administrador del parqueadero, decirle cuántos puestos libres quedan.
Restricción: El manejo del tiempo en la aplicación se debe simular avanzando manualmente la hora vigente del parqueadero.

¿Cuáles son los tres aspectos que debe identificar del enunciado anterior para especificar el problema?
a.

b.

c.

Enuncie los requerimientos:

Requerimiento:	R1 -
Resumen:	
Entradas:	
Resultados:	

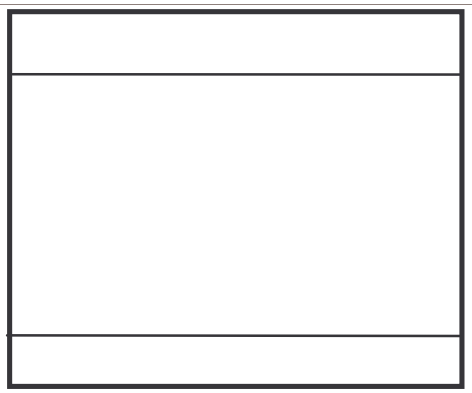
Requerimiento:	R2 -
Resumen:	
Entradas:	
Resultados:	

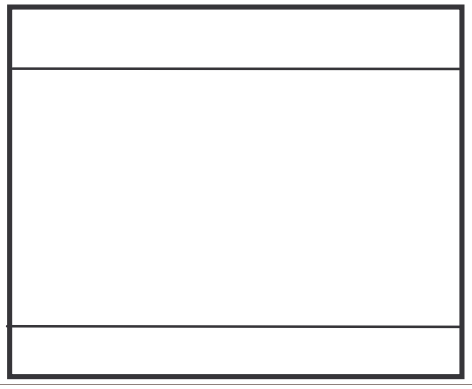
Requerimiento:	R3 -
Resumen:	
Entradas:	
Resultados:	

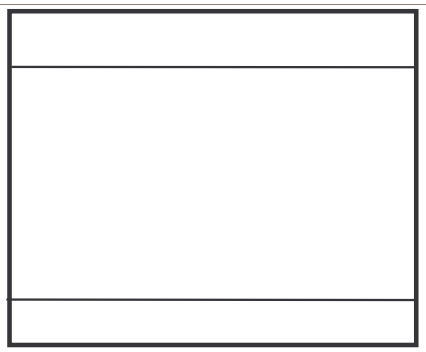
Requerimiento:	R4 -
Resumen:	
Entradas:	
Resultados:	

Describir el Mundo del Problema	*Observe el mundo del problema del parqueadero y resuelva los siguientes puntos*
1. Identifique las entidades del mundo y descríbalas brevemente	

2. Describa las características de cada entidad y represéntelo en UML

Atributo	Valores Posibles	Diagrama UML
		

Atributo	Valores Posibles	Diagrama UML
		

Atributo	Valores Posibles	Diagrama UML
		

3. Dibuje las clases en UML (omita atributos y métodos) y las asociaciones que existen entre ellas

Descomponer y asignar responsabilidades	*Siga las siguientes actividades para designar adecuadamente los métodos de las clases*

1. Para cada requerimiento funcional, haga una descomposición en subproblemas. Enumérelos en una lista e identifique la clase responsable de resolver cada una de ellos

R1 –

R2 –

R3 –

R4 –

2. A partir del análisis anterior, indique para cada clase la lista de los principales métodos que debe tener, descríbalos brevemente indicando función, salida y retorno, e identifique de que tipo son (constructores, modificadores o analizadores)

Clase:

Nombre de método	Descripción	Tipo

Clase:

Nombre de método	Descripción	Tipo

Clase:

Nombre de método	Descripción	Tipo

3. Describa para los métodos principales las precondiciones, postcondiciones, excepciones a disparar. Dé además la firma del método

Clase:	Método
Precondiciones	Sobre el objeto:
	Sobre los parámetros:
Postcondiciones	Sobre el objeto:
	Sobre el retorno:
Excepciones:	
Signatura:	

Clase:	Método
Precondiciones	Sobre el objeto:
	Sobre los parámetros:

Postcondiciones	Sobre el objeto:
	Sobre el retorno:
Excepciones:	
Signatura:	

Clase:	Método
Precondiciones	Sobre el objeto:
	Sobre los parámetros:
Postcondiciones	Sobre el objeto:
	Sobre el retorno:
Excepciones:	
Signatura:	

Construir las clases	*Construya las clases en Java siguiendo los siguientes pasos*

1. Escriba las declaraciones para cada una de las clases identificadas. Incluya atributos y constantes si es necesario.

```
public class
{




}
```

```
public class
{




}
```

```
public class
{




}
```

2. Genere expresiones interesantes en cada una de las clases: a partir de un enunciado lógico para el contexto del problema, escriba la expresión que lo representa. Indique en la descripción la clase en la que se desarrolla.

Descripción	Expresión

3. Escriba los métodos constructores para cada una de las clases

```
public
{




}
```

```
public
{




}
```

```
public
{




}
```

4. Desarrolle los siguientes métodos que involucran instrucciones condicionales

En la clase Carro, decir si la placa del carro comienza con vocal

```
public boolean placaConVocal( )
{




}
```

En la clase Puesto, retornar el número de puesto doblado por 2 si éste es 150,300 ó 450, y retornar el número triplicado en el resto de casos

```
public int placaAumentada( )
{




}
```

En la clase Parqueadero, dado un número de puesto, retornar 1 si el puesto es impar y está ocupado, 2 si es par y está ocupado y 3 si está desocupado

```
public int estadoPuesto( int numero )
{




}
```

5. Desarrolle los siguientes métodos de la clase Parqueadero que involucran instrucciones repetitivas, identificando apropiadamente el patrón correspondiente

Calcular el total de puestos ocupados

Patrón:	`public int totalPuestosOcupados( )` `{`
Variante:	
Decisiones:	`}`

Saber si existe un carro cuya placa comience con la letra dada

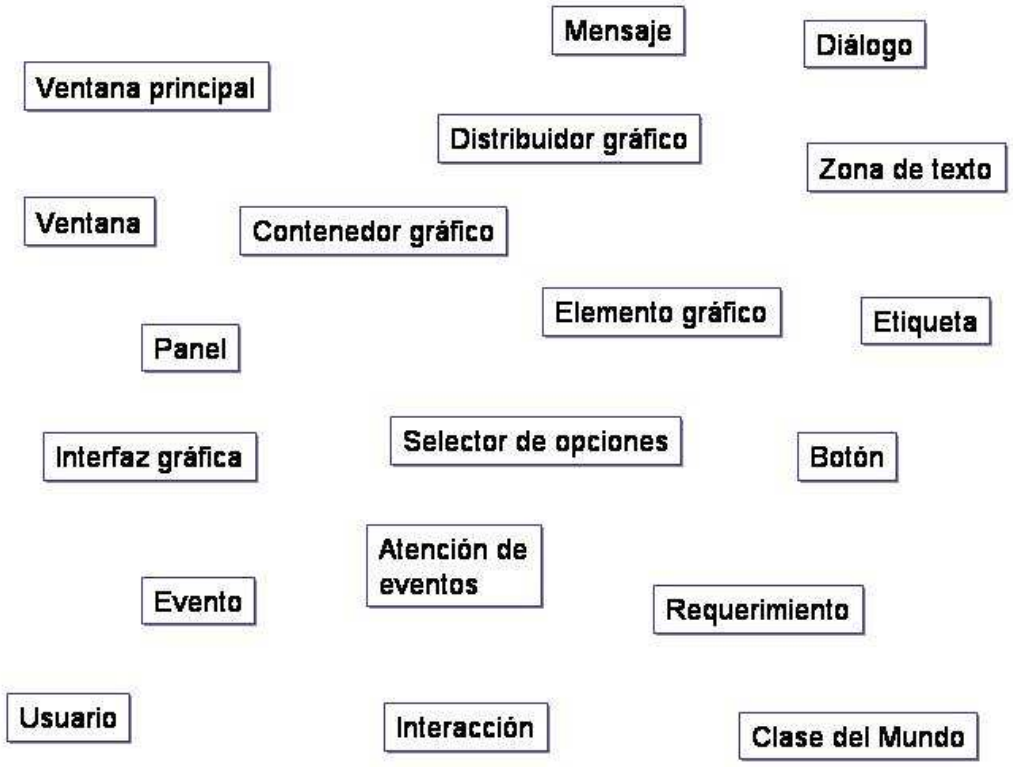
Patrón:	`public boolean existePlacaIniciaCon( char letra )` `{`
Variante:	
Decisiones:	`}`

Actualizar la hora de llegada de cada carro con el promedio de horas de los carros que llegaron antes que él	

Patrón:	`public void actualizarHoraLlegada( )`
	`{`
	
	`}`

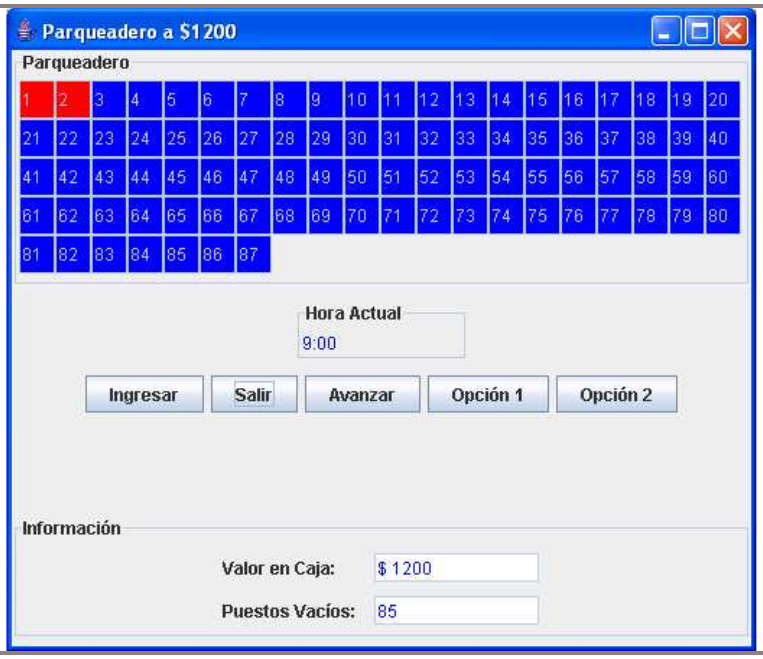
Variante:	
Decisiones:	

Construir mapa conceptual sobre interfaces gráficas	*Identifique relaciones entre los siguientes conceptos, conectándolos con flechas nombradas*

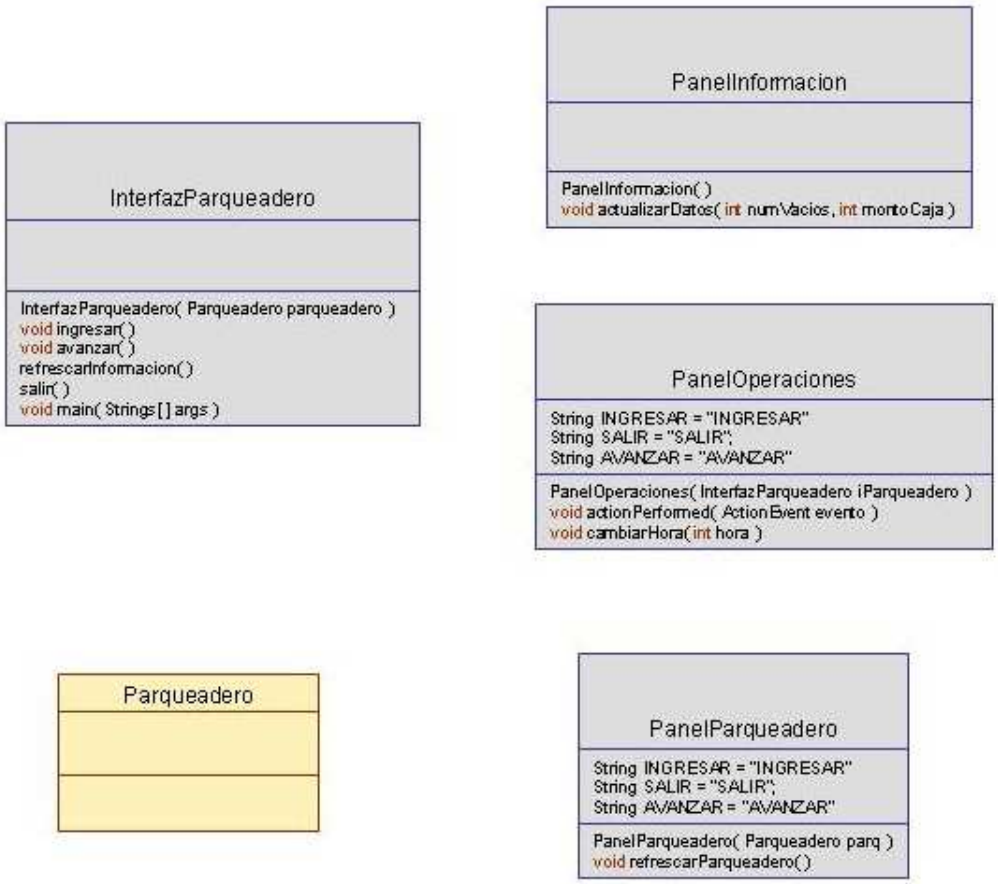


Describir la interfaz gráfica	*Describa los elementos de la interfaz gráfica siguiendo los siguientes pasos*

1. Observe la interfaz gráfica del ejemplo del parqueadero e identifique señalando sobre la imagen todos los elementos gráficos que tiene (botones, campos de texto, ventanas, paneles, diálogos, etc.)



2. Complete el siguiente diagrama de clases de la interfaz agregando los estereotipos y las asociaciones entre ellas. . Utilice los estereotipos para indicar sí es un JFrame, JPanel, JDialog y adicione “-Listener” si la clase tiene el manejo de eventos



¿En qué clase encuentro los requerimientos resueltos?

3. Complete los siguientes constructores de clase. Suponga que los atributos y constantes que requiera para hacerlo ya han sido declarados

```
public InterfazParqueadero(    )
{









}
```

```
public PanelInformacion(    )
{









}
```

```
public PanelOperaciones(     )
{









}
```

```
public PanelParqueadero(     )
{









}
```

4. Resuelva los requerimientos de la aplicación. Revise los diagramas de clases para saber que métodos puede utilizar

```
public void avanzar(  )
{









}
```

```
public void ingresar(  )
{









}
```

```
public void salir(  )
{









}
```