UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ciencias

Introducción a las Ciencias de la Computación

Práctica 4: Clases y objetos.

Profesora: Amparo López Gaona
Ayudante: Ramsés Antonio López Soto
Ayudantes de Laboratorio:
Adrián Aguilera Moreno
Kevin Jair Torres Valencia

Objetivos

El objetivo de esta práctica es que el alumno sea capaz de crear una clase a partir de la definición de un problema.

Introducción

La parte medular de la programación orientada a objetos está formada por las clases, pues de ellas se derivan los objetos. En toda clase se define tanto la estructura como el comportamiento que tendrán sus objetos. La parte estructural de los objetos se define mediante la declaración de datos, éstos pueden ser de cualquier tipo definido por el lenguaje. El comportamiento de los objetos se modela mediante métodos, y es sólo mediante éstos que se puede asignar, alterar y conocer el estado de un objeto.
Un método es el conjunto de instrucciones que se deben realizar al llamar a ejecutar tal método, es decir, al enviar el mensaje correspondiente. En las clases se tienen distintos métodos, los más comunes son:

- `Constructores`. Su nombre es igual al de la clase y su propósito es asignar un estado inicial a los objetos. Este método se llama implıcitamente al utilizar el operador `new`.
- `Modificadores`. Tienen el propósito de modificar el valor de los atributos privados de los objetos. En su forma más simple, reciben como parámetro el nuevo valor para el atributo correspondiente, por tanto, el tipo del parámetro debe coincidir con el tipo del atributo, o bien, ser de tipo compatible. Estos métodos no devuelven valor alguno. El nombre de estos métodos suele empezar con la palabra asignar, seguida del nombre del atributo iniciado con letra mayúscula.
- `Métodos de acceso`. Permiten recuperar el valor de un atributo de la estructura. No reciben parámetros y el valor que devuelven es del tipo definido en el atributo. Generalmente su nombre empieza con la palabra obtener seguido del nombre del atributo. Si se especifica que un método va a devolver un valor se debe usar en el cuerpo del mismo la instrucción return, seguida de una expresión cuyo valor es el que devolverá el método que la contiene.

Resulta de gran importancia y utilidad el que una clase esté documentada. La documentación de la clase en `Java` puede generarse como archivos `HTML` mediante el uso del programa javadoc. Para ello en el programa se deben incluir comentarios entre los sımbolos `/**` y `*/` abarcando todas las lıneas necesarias e incluir ciertas etiquetas. Las etiquetas más utilizadas son:

- `@author` *nombre*. Para especificar el nombre del autor del programa.
- `@param` *nombre descripción*. Para describir cada parámetro de un método.
- `@return` *descripción*. Para especificar el valor que devuelve un método.
- `@see` *referencia*. Para especificar que en el código de esa clase se hace referencia a objetos de otras clases.

- `@version` *descripción.* Para especificar la versión de la clase; ésta se ha usado para incluir la fecha de realización de la clase.
- `@throws` *clase descripción.* Para especificar la clase de excepción que se puede disparar en ese método y en qué situación.

Para utilizar esta herramienta, la clase debe estar precedida de la palabra public. Desde el sistema operativo se debe teclear la instrucción:

```
javadoc -d nombreDeLaCarpeta nombreDePrograma.java
```

con lo cual se genera una serie de archivos con la documentación, en particular uno con el mismo nombre de la clase y extensión html.

Desarrollo

1. Escribe una clase `Tarjeta.java` que modela una tarjeta de débito con los siguientes atributos:

- `numeroTarjeta` que se compone por un `int` de 16 dígitos.
- `fechaCaducidad` que debe estar en el formato `dia-mes-anio`.
- `titular` que es el nombre del titular de la cuenta.
- `emisor` que corresponde al banco que emite la tarjeta.
- `cvv` que corresponde a un código de 3 dígitos numéricos.

Debes incluir 3 constructores:

- Por omisión.
- Por parámetros.
- Por copia.

También debes escribir los respectivos métodos de acceso y modificación con las siguientes nomenclaturas para sus firmas: `obtenerParametro` y `asignarParametro`, donde `Parametro` se refiere al atributo que se esta accediendo o modificando.

Además, debes sobreescribir los métodos `toString()` y `equals(Tarjeta t)`. La clase debe estar completamente documentada y debes generar los archivos resultantes de utilizar `javadoc`.

2. Escribe una clase `Billete.java` que modelea un billeta en varias denominaciones con los atributos:

- `denominacionBillete` que debe ser de tipo `int` y representa el valor del billete en pesos mexicanos.
- `imagenFigurativa` que representa la figura del personaje o personajes que representan el billete.
- `numeroSerie` que debe contener 11 números y letras que validan al billete como único.

Debes incluir 3 constructores:

- Por omisión.
- Por parámetros.
- Por copia.

También debes escribir los respectivos métodos de acceso y modificación con las siguientes nomenclaturas para sus firmas: `obtenerParametro` y `asignarParametro`, donde `Parametro` se refiere al atributo que se esta accediendo o modificando.

Además, debes sobreescribir los métodos `toString()` y `equals(Billete b)`. La clase debe estar completamente documentada y debes generar los archivos resultantes de utilizar `javadoc`.

3. Crea una clase `Cartera.java` que modela una cartera con capacidad máxima de:

- Puede contener 4 billetes, estos deben ser de al menos dos denominaciones distintas. Como atributo usa `private Billete b1, b2, b3, b4;`
- Contiene al menos 2 tarjetas y máximo 3. Puedes incluir el atributo `private Tarjeta t1, t2, t3;`
- `material` que corresponde al material del que esta fabricada.
- `apartados` que corresponde al número de apartados disponibles en la tarjeta.

Debes incluir 3 constructores:

- Por omisión.
- Por parámetros.
- Por copia.

También debes escribir los respectivos métodos de acceso y modificación con las siguientes nomenclaturas para sus firmas: `obtenerParametro` y `asignarParametro`, donde `Parametro` se refiere al atributo que se esta accediendo o modificando.

Además, debes sobreescribir los métodos `toString()` y `equals(Cartera c)`. La clase debe estar completamente documentada y debes generar los archivos resultantes de utilizar `javadoc`.

4. Escribe una clase que implemente un método `main` que cree 3 carteras con sus respectivos atributos y dos de ellas sean iguales pero los elementos contenidos en ella no lo sean. Verifica con su respectivo método equals esta propiedad.

Además, modifica los valores `null` que hayas creado cuando invocas los constructores de dichas carteras.

Finalmente, imprime la representación de las 3 carteras.

Formato de Entrega

1. Las prácticas serán entregadas de forma individual.
2. Cada práctica (sus archivos y directorios) deberá estar contenida en un directorio llamado `apellidoPaterno_nombre_pX`, donde X es el número de la práctica.
   Por ejemplo: `aguilera_adrian_p4`
3. NO incluir los archivos .class dentro de la carpeta.
4. Los archivos de código fuente deben estar documentados.
5. Se pueden discutir y resolver dudas entre los integrantes del grupo. Pero cualquier práctica plagiada total o parcialmente será penalizada con cero para los involucrados.
6. La práctica se debe subir al Github Classroom correspondiente.
7. La entrega en classroom debe contener el link HTTPS y SSH de su repositorio y es lo único que se debe entregar.
8. El horario y día de entrega se acordará en la clase de laboratorio y no deberá sobrepasar 2 clases de laboratorio.