Unidad 1

DIPLOMATURA EN PROGRAMACION JAVA

Ejercicios

Capítulo 2

Visibilidad de Variables y Expresiones

Capítulo 2

Ejercicio 1

En este ejercicio se investigan las variables de referencia de Java, la creación de objetos, la asignación de variables de referencia, el manejo de cadenas a que almacenan valores numéricos en formato de caracteres y las conversiones de tipos para el manejo de cadenas como números utilizando clases de envoltorio.

Utilizando la clase que se le provee cuyo nombre es MiPunto, crear la clase TesteoMiPunto que debe hacer lo siguiente:

1. Declarar dos variables de tipo MiPunto llamadas comienzo y fin. Asignar a cada variable un objeto nuevo.
2. Inicializar los valores x e y del objeto comienzo con la cadena de caracteres 10.
3. Inicializar los valores x e y de fin con las cadenas de caracteres 20 y 30 respectivamente.
4. Mostrar por consola los valores recientemente almacenados. Por ejemplo, para el valor x de comienzo la instrucción sería:

```
System.out.println("El valor x del comienzo es " + comienzo.x);
```

5. Compilar y ejecutar el programa
6. Declarar una nueva variable del tipo MiPunto y llamarla otroPunto
7. Asignarle a otroPunto el contenido de la variable de referencia fin
8. Mostrar por consola los valores x e y de fin y otroPunto
9. Asignar con las cadenas de caracteres 47 y 35 a las variables x e y del objeto al que referencia otroPunto
10. Mostrar por consola los valores x e y para comienzo, fin y otroPunto
11. Corroborar que la salida sea como la siguiente

```
El valor x del comienzo es 10
El valor y del comienzo es 10
El valor x del fin es 20
El valor y del fin es 10
El valor x del otroPunto es 20
El valor y del otroPunto es 10
El valor x del fin es 20
El valor y del fin es 10
El valor x del otroPunto es 40
El valor y del otroPunto es 50
El valor x del fin es 40
El valor y del fin es 50
El valor x del otroPunto es 47
El valor y del otroPunto es 50
El valor x del fin es 47
```

```
El valor y del fin es 50
```

12. Realizar las operaciones ***aritméticas en base a las conversiones necesarias*** para que las salidas por pantalla se vean como se muestra a continuación:

```
El valor x del comienzo es 10
El valor y del comienzo es 10
El valor x del fin es 20
El valor y del fin es 10
El valor x del otroPunto es 20
El valor y del otroPunto es 10
El valor x del fin es 20
El valor y del fin es 10
El valor x del otroPunto es 40
El valor y del otroPunto es 50
El valor x del fin es 40
El valor y del fin es 50
El valor x del otroPunto es 47
El valor y del otroPunto es 50
El valor x del fin es 47
El valor y del fin es 50
----- Operaciones aritméticas ------
El valor x del comienzo es 10
El valor y del comienzo es 10
El valor x del otroPunto es 20
El valor y del otroPunto es 30
El valor x del fin es 20
El valor y del fin es 30
El valor x del otroPunto es 47
El valor y del otroPunto es 45
El valor x del fin es 47
El valor y del fin es 45
```


Ejercicio 2

Evaluar las expresiones aritméticas, lógicas y de comparación

1. Edite la clase TesteoMiPunto y en la función main agregar las siguientes expresiones al final del código, declarando las variables auxiliares que considere necesarias:

    1. Incrementar en uno la variable x de comienzo con el operador unario de pre incremento y mostrar por pantalla
    2. Decrementar en uno la variable y de comienzo con el operador unario de post decremento y mostrar por pantalla
    3. Escribir con notación reducida (ejemplo, a+=4;) la suma de la variable x de fin con la variable x de comienzo y asignarla a x de fin. Mostrar por pantalla

4. Suponiendo que otroPunto.x, fin.x y otroPunto.x representan variables enteras de los respectivos objetos y puede operar con ellas como si fueran variables normales, mostrar por pantalla el resultado de la siguiente expresión y explicar lo que sucede

   otroPunto.x=++fin.x+comienzo.x---otroPunto.x;

5. Almacenar el estado de cada variable de la expresión en sus respectivos objetos en formato de cadena. Mostrar el valor almacenado en otroPunto después de la operación aritmética
6. Tomar este último resultado y desplazar 5 bits a izquierda y volverlo a almacenar en otroPunto.x. Mostrar el valor obtenido en pantalla.
7. Comparar con el operador ternario (?:) si otroPunto.x es más grande que fin.x y mostrar el mayor por pantalla en una sola instrucción (¿qué pasa si son iguales?)
8. Verificar que la salida sea

```
El valor x del comienzo es 10
El valor y del comienzo es 10
El valor x del fin es 20
El valor y del fin es 10
El valor x del otroPunto es 20
El valor y del otroPunto es 10
El valor x del fin es 20
El valor y del fin es 10
El valor x del otroPunto es 40
El valor y del otroPunto es 50
El valor x del fin es 40
El valor y del fin es 50
El valor x del otroPunto es 47
El valor y del otroPunto es 50
El valor x del fin es 47
El valor y del fin es 50
------ Operaciones aritméticas ------
El valor x del comienzo es 10
El valor y del comienzo es 10
El valor x del otroPunto es 20
El valor y del otroPunto es 30
El valor x del fin es 20
El valor y del fin es 30
El valor x del otroPunto es 47
El valor y del otroPunto es 45
El valor x del fin es 47
El valor y del fin es 45
------- Nuevas Operaciones -------
El valor x del comienzo es 11
El valor y del comienzo es 9
El valor x del fin es 58
El valor x del otroPunto es 59
```

```
El valor x del otroPunto es 384
384
```

Ejercicio 3

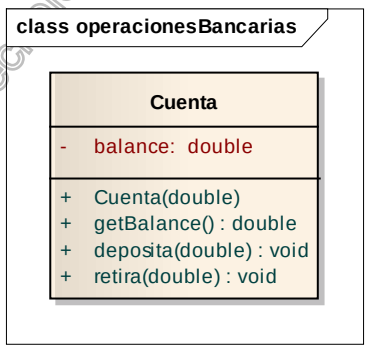
Crear un método en la clase MiPunto que calcule la diferencia al restar el valor de comienzo al de fin para las variables x e y de manera que el método para realizar eso reciba como parámetro una referencia a un objeto de tipo MiPunto. El método, cuando se ejecuta, guarda el resultado en los atributos que pertenecen al objeto que se utilizó para invocarlo. Crear una función main y probar el correcto funcionamiento del método.

Ejercicio 4

Este ejercicio introduce al proyecto de operaciones bancarias, al cual se utilizará en módulos futuros. El mismo consiste de un banco con diferentes clientes con distintas cuentas cada uno y reportes. Estas clases se reutilizarán en los sucesivos ejercicios

Se deberá crear una versión simple de la clase Cuenta la cual se colocará en el paquete operacionesBancarias. Un programa de prueba que se encuentra en la clase PruebaOperacionesBancarias, se escribió en el paquete por defecto para crear una única cuenta. Este inicializa en balance de la cuenta y realiza algunas transacciones simples. Por último el programa muestra el balance resultante.

1. Abrir el proyecto del directorio Módulo 2\Ejercicio 4
2. Crear el paquete operacionesBancarias
3. Crear la clase Cuenta que refleje el siguiente diagrama UML



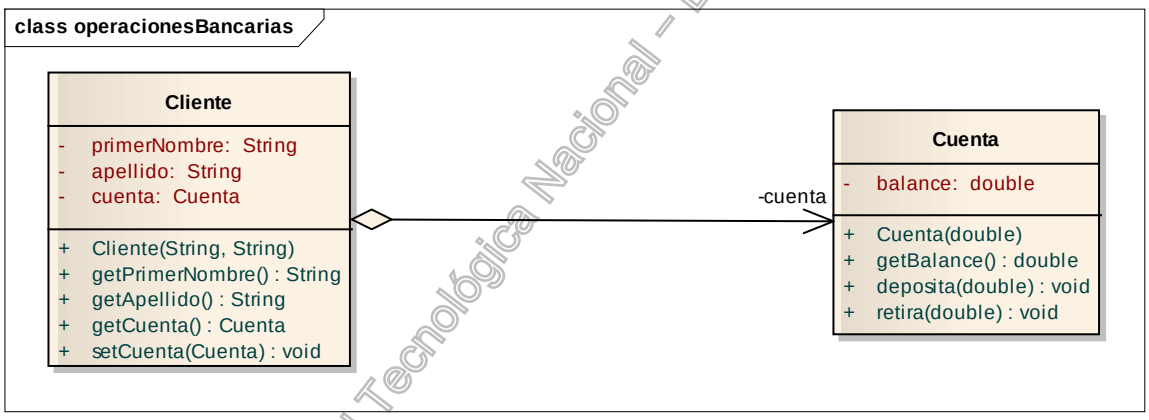
4. Declarar como privado un atributo: balance. Este atributo almacena el balance en curso de la cuenta bancaria
5. Declara un constructor público que recibe un solo parámetro, el balance inicial, el cual se asignará al balance en curso
6. Declarar un método público getBalance, que retorna el balance actual

7. Declarar un método público deposita, que agrega la cantidad que recibe como parámetro al balance
8. Declarar un método público retira que extrae la cantidad que se recibe como parámetro del balance actual
9. Leer el código de la clase PruebaOperacionesBancarias.
10. Compilar y ejecutar las clases
11. Verificar que la salida sea igual que la que se muestra:

```
Creando una cuenta con 500.00 de balance.
Retira 150.00
Deposita 22.50
Retira 47.62
La cuenta tiene un balance de 324.88
```

Ejercicio 5

En este ejercicio se expandirá el proyecto de operaciones bancarias agregándole la clase Cliente, donde el cliente tendrá un objeto del tipo cuenta, como muestra el siguiente diagrama UML



1. Abrir el proyecto del directorio Módulo 2\Ejercicio 5
2. Crear la clase Cliente en el paquete recién creado de manera que respete el gráfico UML previamente expuesto
3. Declarar tres atributos privados: cuenta, primerNombre y apellido
4. Declarar un constructor público que reciba dos parámetros (p y a) para inicializar los atributos de los objetos
5. Declarar dos métodos de acceso públicos para los atributos del objeto de manera de obtener el primer nombre y el apellido
6. Declarar el método setCuenta para asignar el atributo cuenta
7. Declarar el método getCuenta para recuperar el atributo cuenta
8. Ejecutar el método main de la clase PruebaOperacionesBancarias y verificar que la salida sea la siguiente

```
Creando una cuenta con 500.00 de balance.
Retira 150.00
Deposita 22.50
Retira 47.62
La cuenta tiene un balance de 324.88
Cliente [Perez, Juan] tiene un balance de 324.88
```