PRÁCTICA 6
Conceptos básicos del lenguaje Java

1. Implemente la clase Lamparita, que sirva para representar el estado de encendido de una lamparita (encendido o apagado). Defina, asimismo, dos métodos que permitan encender y apagar la luz de la lamparita y otro que indique en qué estado se encuentra. La lamparita inicialmente está apagada.

2. Dada la siguiente definición de clase:

```
public class Archivo {
   String nombre;
   int longreg;

   void init (int r) {
      longreg = r;
   }
};

Archivo f = new Archivo();
```

   ¿Qué sucede cuando se aplica `f.init(80)`?

3. Indique cuál es la salida del siguiente programa:

```
public class Punto {
   public int x;
   public int y;
   public Punto(int a, int b) {x = a; y = b;}
   public Punto(int z) {this(z, z);}
}

public class Prueba {
   public static void main(String[] args) {
      Punto p = new Punto(25);
      System.out.println("x = " + p.x + " y = " + p.y);
   }
}
```

4. ¿Cuál es el constructor que se invoca en la siguiente declaración?

```
public class Prueba {
   private int num;
   public Prueba() {num = 0;}
   public Prueba(int n) {num = n;}
   public int valor() {return num;}
}

Prueba prueba = new Prueba();
```

5. Se dispone de los siguientes métodos de una clase dada:

```
public void f(char c);
public void f(int i);
```

```
public void f(double d);
```

¿A cuál corresponde cada una de las siguientes invocaciones?

- `f(72.25);`
- `f(0);`
- `f('z');`

6.

a) Implemente la clase Hora que contenga miembros datos separados para almacenar horas, minutos y segundos. Un constructor inicializará estos datos en 0 y otro a valores dados. Una función miembro deberá visualizar la hora en formato *hh*:*mm*:*ss*. Otra función miembro sumará dos objetos de tipo hora, retornando la hora resultante. Realizar otra versión de la suma que asigne el resultado de la suma en el primer objeto hora.

b) Programar un procedimiento `main()`, que cree dos horas inicializadas y uno que no lo esté. Se deberán sumar los dos objetos inicializados, dejando el resultado en el objeto no inicializado. Por último, se pide visualizar el valor resultante.

7. Implemente la clase Empleado, que contenga como atributos miembros el número y nombre de empleado, y los siguientes métodos miembros:
   a) `getNumero()`: Obtiene el número del empleado dado
   b) `getNombre()`: Obtiene el nombre del empleado dado
   c) `setNumero(int num)`: Modifica el número del empleado dado
   d) `setNombre(String nom)`: Modifica el nombre del empleado dado
   e) `verDatos()`: Visualiza los datos del empleado por standard output

   Luego escribir un método que cree varios empleados, los complete con datos de empleados y luego visualice los datos de los mismos.

8. Implemente la clase Punto (pares de coordenadas de tipo float x, y). Defina constructores y métodos para asignar valores a las coordenadas de los puntos, retornar el valor de cada coordenada, y sumar dos puntos, retornando su resultado. Definir un método booleano de igualdad entre dos puntos.

9. Implemente la clase Vector3D (ternas de coordenadas de tipo float x, y, z). Defina constructores y métodos para asignar valores a las coordenadas de los vectores3D, retornar el valor de cada coordenada, y sumar dos vectores3D, retornando su resultado. Definir un método booleano de igualdad entre dos vectores3D. Implementar esta clase a partir de la implementación de la clase Punto.

10. Implemente la clase Complejo (números complejos). Defina constructores y el método de multiplicación de complejos. Además programe tres funciones suma(), una que reciba dos números de tipo `int`, otra que reciba dos números de tipo `float`, y otra que reciba dos números complejos.

11. La coordenadas propias de la clase anterior Punto representan coordenadas rectangulares (coordenadas x, y). Estas coordenadas se pueden escribir como coordenadas polares (radio, ángulo) de la siguiente manera:

    x = radio * cos(ángulo)
    y = radio * sen(ángulo)

Implemente la clase Polar, cuyos miembros sean el radio y ángulo, y que posea métodos de obtención de cada coordenada polar, y otro que convierta las coordenadas polares en rectangulares. Programar un método que permita sumar coordenadas polares. Realizar un método de prueba para la clase.

12. Implemente la clase Fecha, que permita representar una terna de día, mes y año, todos de tipo entero. Programar un método que determine si una fecha es mayor a otra. Programar la sobrecarga del método `toString()`.

13. Implemente la clase Fraccion con las operaciones aritméticas comunes (suma, resta, multiplicación y división), dos constructores, y el método `toString()`.

14. Implemente la clase Ecuacion, que sea capaz de construir una ecuación de segundo grado (con sus componentes de tipo flota), y de calcular las raíces reales de la ecuación.

15. Implemente la clase Potencia, que permita crear objetos de la forma (base -float-, exponente -natural-). Programar un método `evaluar()`, que retorne el resultado de la evaluación de la potencia, utilizando recursión.

16.
    a) Implemente las clases NumeroBinario y NumeroDecimal . La primera construye números binarios expresados como sucesiones de caracteres `String` de dígitos `'0'` y `'1'`, y la segunda construye números naturales en notación decimal. Programar métodos `aDecimal()` y `aBinario()`, que permitan convertir números de binarios a decimal, y viceversa (retornando los resultados como objetos de la otra clase).
    b) Programar una variante de las clases anteriores con sus métodos, si éstos fueran de tipo `static`. Justificar.

17. Implemente una clase Monedero que permita gestionar la cantidad de dinero que una persona dispone en un momento dado. La clase deberá tener un constructor que permitirá crear un monedero con una cantidad de dinero inicial y deberá definir un método para meter dinero en el monedero, otro para sacarlo y finalmente, otro para consultar el disponible; solo podrá conocerse la cantidad de dinero del monedero a través de este último método. Por supuesto, no se podrá sacar más dinero del que haya en un momento dado en el monedero.

18. Implemente una clase CuentaCorriente, que permita llevar el control del estado de una cuenta corriente. La cuenta corriente está caracterizada esencialmente por su saldo, y sobre ella se pueden realizar los siguientes tipos de operaciones:
    - `saldo()`: Retorna el saldo de la cuenta corriente (puede ser negativo).
    - `deposito(float imp)`: Deposita un importe dado a la cuenta corriente.
    - `extraccion(float imp)`: Extrae un importe dado de la cuenta corriente.
    - `cantidadOperaciones()`: Retorna la cantidad de operaciones válidas realizadas.
    - `cantidadExtraccionesInvalidas()`: Retorna la cantidad de extracciones en las que se intentó extraer más dinero del que existía en la cuenta corriente.

19. Implemente una clase Microondas que permita cocinar una comida. El microondas posee una puerta que puede estar abierta o cerrada, y un contenido que puede tener una comida o estar vacío. Se puede insertar una comida (si la puerta está abierta, y no hay otra comida adentro), retirar una comida (si la puerta está abierta y hay una comida adentro). También se puede iniciar una cocción con un nivel de intensidad de cocción y una cantidad de segundos de cocción, si la puerta está cerrada, y hay una comida adentro. También se puede finalizar la cocción (cuando se acaba su tiempo), lo cual la comida cocinada tendrá tantos puntos adicionales de cocción como el nivel de intensidad de la cocción finalizada multiplicado por la cantidad de segundos de esa cocción. Cuando una cocción se encuentra en curso, la puerta no se puede abrir. Lo que sí se puede hacer es abortar una cocción a una cantidad de segundos faltantes, si existe una cocción en curso. El efecto de esto último sería cocinar la comida por sólo el tiempo parcial en que estuvo cocinándose (el tiempo de la cocción del inicio menos el tiempo que faltaba al abortar) y abrir la puerta.
    También se solicita que se pueda mostrar el estado completo del microondas en cualquier momento, incluyendo el de la comida que puede estar dentro. La comida se caracteriza por poseer un nombre y una cantidad de puntos de cocción.

20.
    a) Explique qué ocurre en el entorno de una porción de código si una expresión propia de ese código, de un tipo básico de Java se pasa como parámetro a un método invocado, y esta expresión es modificada en el método invocado. ¿En qué estado queda la expresión en la porción de código original?
    b) Idem anterior, pero lo que se pasa como parámetro no es una expresión de un tipo básico, sino un objeto de una clase. ¿Qué ocurre si dentro del método invocado se le aplican métodos que modifican su estado interno?
    c) Idem b), pero preguntando si dentro del método invocado se le cambia la identidad a ese objeto (a la variable que representa ese objeto se le asigna otro objeto distinto)?

21. Convierta todos los tipos definidos anteriormente como TADs no recursivos, a clases del lenguaje orientado a objetos Java.