Universidad Nacional Autónoma de México
Posgrado en Ciencia e Ingeniería de la Computación

Programación Avanzada
Semestre 2023-1
Tarea Pre Examen

1. Ejercicios Teóricos

1. ¿Qué elementos definen a un objeto?

   a) Su cardinalidad y su tipo

   b) Sus atributos y sus métodos

   c) La forma en que establece comunicación e intercambia mensajes

   d) Su interfaz y los eventos asociados

   e) Todas lo definen

2. ¿Qué significa instanciar una clase?

   a) Duplicar una clase

   b) Crear un objeto a partir de la clase

   c) Eliminar una clase

   d) Heredar de una clase

   e) Conectar dos clases entre sí

3. Es el área de memoria donde viven las variables locales

   a) The Heap

   b) The Stack

   c) The Garbage Collector

   d) The Vector

   e) The Phantom Zone

4. ¿Una clase que hereda de una clase abstracta puede ser no abstracta?

   a) Si, si implementa algunos métodos abstractos de la superclase

   b) Si, si implementa todos los métodos abstractos de la superclase

   c) Sí, siempre que no se antoponga la palabra reservada *abstract*

   d) No, en java siempre debe ser abstracta o hay un error de compilación.

   e) No, porque se genera un error en tiempo de ejecución.

5. A una variable que se declara en el cuerpo de una clase con el modificador *static*, se le conoce como:

   a) Variable de instancia

   b) Variable de clase

   c) Variable local

   d) Variable dinámica

   e) Ninguna de las anteriores

6. ¿Cuál de los siguientes códigos está relacionado con la herencia en Java

   a) `public class Perro implements Animal`

   b) `public class Perro inherit Animal`

   c) `public class Perro is an Animal`

   d) `public class Perro extends Animal`

   e) Ninguna de las anteriores

7. ¿Cuál de las lineas en el código *UnaLineaMal.java* nos arroja un error en compilación. (ver código 1).

   a) Línea 3

   b) Línea 4

   c) Línea 5

   d) Línea 6

   e) Todas son correctas

8. ¿Cuál es el resultado de compilar y ejecutar el código *TryCatch.java* (ver código 2). Justifica tu respuesta

   a) 0

   b) infinity

   c) Se imprime el mensaje "ERROR"

   d) Error en tiempo de compilación

   e) Error en tiempo de ejecución

9. Dado lo siguiente, selecciona la(s) opcion(es) correcta(s).
   A y E son clases.
   B y D son interfaces.
   C es una clase abstracta.

   a) La clase F implementa B y D.

   b) La clase F implementa B.

   c) La clase F extiende A y E

   d) La clase F extiende E.

   e) La clase F implementa B y C.

10. De la siguiente lista ¿Cuáles son identificadores válidos en Java?

    a) $aluda

    b) Saluda

    c) 5aluda

    d) @saluda

    e) _saluda

11. Selecciona la(s) característica(s) que describa(n) mejor a las excepciones comprobadas (checked exception) en java.

    a) Son todas las descendientes de RuntimeException.

    b) Su captura o declaración es obligatoria.

    c) No están sujetas a captura, por lo que podemos ejecutar un programa sin tomarlas en cuenta.

    d) Son todas las descendientes de Exception excepto las de RuntimeException.

    e) Son excepciones que están fuera de nuestro alcance como errores repentinos de hardware y no pueden ser tratadas.

2. Preguntas Abiertas

1. Observe la clase *EjercicioHeap.java* (ver código 3). Al momento de alcanzar la línea 18 (//continua), algunos objetos y algunas variables de referencia ya habrán sido creadas ¿Cuál es el estado de estas variables con respecto a los objetos?

2. Sean las clases A,B y Test con el contenido que se muestra en el código 4. Sin modificar la clase A ni la clase Test, agregue en la clase B lo mínimo necesario con el fin de que el programa compile y ejecute exitosamente. Explique detalladamente su propuesta de solución.

3. Escribe con tus propias palabras de manera clara y concisa ¿Qué es la sobrecarga (overload) y la sobreescritura (override) en java?

4. Investiga y describe cuál es la diferencia entre String y StringBuffer. Da algunos ejemplos explicados que ayuden a clarificar la idea. (Agrega una hoja para explicación)

3. Ejercicios Prácticos

Instrucciones:

- Genere un solo proyecto en el IDE.
- Emplea paquetes para organizar los ejercicios
- Emplea la convencion camel case para los ejercicios.
- Entregar un archivo comprimido con los códigos empaquetados.

1. Escriba un programa que calcule el promedio de N números. El usuario deberá ingresar por teclado el total de números a promediar y los correspondientes números. El programa debe tener validaciones para evitar errores, bajo ninguna circunstancia debe finalizar abruptamente (Use el manejo de excepciones)

2. Escribe un método que verifique si una sentencia es palíndromo sin emplear ninguna función de la API de java que automátice el proceso, por ejemplo no se puede usar compareTo(), replace(), replaceAll() o cualquier otra función ya existente en el API. El usuario deberá ingresar la palabra o enunciado por teclado. El método debera ignorar los espacios en blanco tanto al inicio, en medio y al final de la oración, así como los signos de puntuacion (puntos, comas.), no habrá distinción entre mayúsculas y minúsculas y devolverá el valor de *true* si es palíndromo o *false* en caso contrario. No ingrese acentos en los enunciados. Pruebe la funcionalidad con una clase *Test* como se ha hecho en tareas previas.

   A continuacion se muestran algunas cadenas de prueba:

   - Oxxo
   - somos no somos
   - s o m o s, no, som..os
   - Alli ves, Sevilla
   - Asi, Maria, raparas a Sara para ir a misa

3. Escribe una clase Persona que contenga 3 atributos: nombre, apellido y edad; usa modificadores y métodos de acceso correctos. Además, Escribe una clase que nos sirva como clase de utileria que contenga el método necesario para comparar a **tres personas** por edad y nos **devuelva** cual **persona** es la mayor. Este método debe poder ser usado sin la necesidad de crear una instancia de esa clase. Finalmente, usa una clase *Test* que contenga el método main para que realices las pruebas necesarias para comprobar la funcionalidad y por último se muestren todos los datos de la persona mayor.

4. Códigos

```java
public class UnaLineaMal {
    public static void main(String[] args) {
        double d, int a;
        int i, j = 0;
        byte b = 127;
        double pi = 3.141592, e = 2.71828;
        System.out.println(pi);
    }
}
```

Código 1: UnaLineaMal.java

```java
public class TryCatch {

    public static void main(String[] args) {
        int x = 0;
        int y = 10;

        try {
            y = y / x;
        }
        System.out.println("El divisor es cero");
        catch (Exception e) {
            System.out.println("ERROR");
        }
    }
}
```

Código 2: TryCatch.java

```java
public class EjercicioHeap {
    private int id;
    public EjercicioHeap(int id) {
        this.id = id;
    }
    public static void main(String[] args) {
        EjercicioHeap[] eh = new EjercicioHeap[5];
        int x = 0;
        while (x < 5) {
            eh[x] = new EjercicioHeap(x);
            x++;
        }
        eh[0] = eh[3];
        eh[4] = eh[1];
        eh[3].id = 7;
        eh[1] = eh[2];
        eh[2] = eh[4];
        // continua
    }
}
```

Código 3: EjercicioHeap.java

```java
public class A {
    private  A() {
        System.out.println("A");
    }
    private A(int i){
        this();
        System.out.println(i);
    }
    protected A(char c){
        this(5);
        System.out.println(c);
    }
}

public class B extends A{
    public B(){
        System.out.println("B");
    }
}

public class Test{
    public static void main(String[] args) {
        B b = new B();
        System.out.println("hola");
    }
}
```

Código 4: A.java B.java Test.java