UNLaM

Escuela de Formación Continua

PROGRAMACIÓN AVANZADA I

1er Parcial

Apellido y Nombre

DNI:

13/05/2022

Calificación:

1) Dado el siguiente código, indique brevemente 3 de los PRINCIPALES ERRORES que ocurren o pueden ocurrir (2pt)

```
public static void main(String[] args) {
        File f = new File("numeros.txt");
        Scanner sc = new Scanner(f);
        int a = sc.nextInt();//nextInt funciona como nextLine, pero para enteros
        int b = sc.nextInt();//nextInt funciona como nextLine, pero para enteros
        System.out.println(a / b);
    }
```

Rta:

a) No se maneja la excepción al abrir el archivo (puede no existir)

b) El archivo puede estar vacío, y daría una excepción al hacer 'sc.nextInt()'

c) No se maneja la posible división por cero

==========

2) Dado el siguiente código, escriba la salida por pantalla al ejecutarlo (2pt)

```
public static void main(String[] args) {
        int[] arr = {15, 10, 25, 40, 32};
        for(int i = 0; i < 5; ++i){
            System.out.println(arr[i]);
            if(i == 3){
                break;
            }
        }
    }
```

Rta: 15 10 25 40 ____ ____ ____ ____ ____

==========

3) Dadas las siguientes clases e interfaces, indique que métodos debería implementar la clase Caballo para que el programa no arroje ERRORES DE COMPILACIÓN (2pt)

```
public abstract class Animal{
    • public abstract void emitirSonido();
    • public void alimentarse(){
    }
}
public interface Corredor {
    • public void correr();
}
public class Caballo extends Animal implements Corredor {
}
```

Rta:

Correr EmitirSonido

==========

Considere el siguiente código para las preguntas 4 y 5

```
public class CustomException extends RuntimeException {

    public CustomException(String message){

        super(message);
    }
}
```

4) ¿Qué función cumple la línea '**`super(message)`**'? (1pt)
   a. Llama al constructor de la clase Super
   b. Llama al constructor de la clase hija, que recibe un parámetro de tipo "String"
   c. Llama al constructor de la clase padre, que recibe un parámetro de tipo "Message"
   d. Llama al constructor de la clase padre, que recibe un parámetro de tipo "String"

5) ¿Cuál sería una opción para atrapar esta excepción en el código? (1pt)
   a. Usando un bloque try with resources
   b. Usando un bloque try / catch, donde el catch reciba una excepción de tipo Exception
   c. Usando un bloque try / catch, donde el catch reciba una excepción de tipo IOException
   d. Usando un bloque if / else, preguntando si la excepción es una instancia de CustomException

==========

6) Dado el siguiente código, ¿qué problema encuentra? Asuma que el archivo "archivo.txt" existe y no está vacío (1pt)

```
public static void main(String[] args) throws FileNotFoundException {

        try (Scanner sc = new Scanner(new File("archivo.txt"))){

            while(sc.hasNextLine()){

                System.out.println("Esta es la proxima linea del archivo");
            }
        }
 }
```

   a. El while queda en un loop infinito
   b. El scanner no se cierra después de usarse
   c. Tiene error de compilación porque no hay un catch para la excepción
   d. Scanner siempre pisa el archivo y crea uno nuevo, por lo que estará vacío

7) Dada la firma de la clase Pato, ¿qué error de compilación tiene? Asuma que todas las clases e interfaces existen (1pt)

```
public class Pato extends Animal, SerVivo implements Nadador, Corredor, Volador {
```

   a. La clase debe ser abstracta, sino no puede heredar de otra clase
   b. La clase no puede implementar más de 2 interfaces
   c. La clase no puede heredar de 2 clases distintas
   d. No tiene ningún error