Tarea 8
Programación avanzada

Emmanuel Peto Gutiérrez

17 de octubre de 2022

1. Ejercicios teóricos

1. ¿Qué elementos definen a un objeto?

   b) Sus atributos y sus métodos

2. ¿Qué significa instanciar una clase?

   b) Crear un objeto a partir de la clase

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

   b) The Stack

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

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

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

   b) Variable de clase

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

   d) `public class Perro extends Animal`

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

   a) Línea 3

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

   d) Error en tiempo de compilación.

   No debe haber código entre la llave que cierra el try y el inicio del catch, pero en este caso hay una línea: `System.out.println(‘‘El divisor es cero’’);`

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.
   d) La clase F extiende E.

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

    a) $aluda
    e) _saluda

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

    b) Su captura o declaración es obligatoria.

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?

Interprétese la notación (id: n) como un objeto de tipo `EjercicioHeap` cuyo atributo id es n. Se tienen los siguientes objetos: (id: 0), (id: 1), (id: 2), (id: 7), (id: 4). Al final, cada casilla del arreglo apunta a un objeto de la siguiente forma:

- eh[0] $\rightarrow$ (id: 7)
- eh[1] $\rightarrow$ (id: 2)
- eh[2] $\rightarrow$ (id: 1)
- eh[3] $\rightarrow$ (id: 7)
- eh[4] $\rightarrow$ (id: 1)

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.

La clase `B` tendría el siguiente cuerpo:

```
public class B extends A{
    public B(){
        super('C');
```

```
        System.out.println("B");
    }
}
```

La clase `A` tiene un constructor protegido que puede ser invocado desde cualquier hijo de `A` y este constructor es el que recibe un caracter. Dentro del constructor de `B` se invoca a ese constructor de `A` con la palabra `super` y se le pasa como argumento el caracter `C` (aunque funciona con cualquier caracter).

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

Sobrecarga. Una sobrecarga se da cuando dos o más métodos tienen el mismo nombre pero la firma cambia. Para que la firma cambie, los tipos de los parámetros o el número de parámetros que reciben deben ser diferentes. Una sobrecarga de dos métodos puede ocurrir en alguno de los siguientes casos:

- Ambos están en la misma clase.
- Un método está en una clase `A` y el otro en una clase `B` que es descendiente de `A`.

Un ejemplo de sobrecarga son los métodos `add` de la clase `LinkedList`. Si los elementos de la lista son de tipo `T`, el método `add(T e)` agrega un elemento al final de la lista, mientras que el método `add(int index, T e)` agrega un elemento en la posición `index` de la lista. Aunque tengan el mismo nombre los métodos son diferentes, pues uno recibe un argumento y el otro recibe dos.

Sobreescritura. Una sobreescritura se da cuando hay dos métodos que tienen la misma firma, el tipo de retorno es el mismo pero uno está en una clase `A` y el otro en una clase `B` que es descendiente de `A`, y los métodos tienen comportamientos diferentes.

Para sobreescribir un método, se debe escribir `@Override` justo arriba de la firma del método en la clase descendiente.

Un ejemplo de sobreescritura es el método `hashCode` en la clase `Integer`. Este método no recibe parámetros y devuelve un `int`. La clase `Integer` es descendiente de `Object`. En la clase `Object`, el método `hashCode` toma la dirección en memoria del objeto y la transforma a un número entero. La clase `Integer` guarda exactamente un número entero y el método `hashCode` devuelve ese número.

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)

La diferencia entre un `String` y un `StringBuffer` es que un objeto de tipo `StringBuffer` tiene métodos para su modificación mientras que uno de tipo String no. Se mostrarán ejemplos de algunos de estos métodos. Para los ejemplos, suponga que se tiene un objeto de tipo `StringBuffer` llamado `sb` cuyo valor es "pedro".

- **append**: concatena un elemento de cualquier tipo al final de la cadena. Si se ejecuta `sb.append(‘‘ perez’’)`, el valor actual de `sb` será “pedro perez”.
- **delete**: Elimina la subcadena entre los índices especificados, incluyendo el primero y excluyendo al segundo. Al ejecutar `sb.delete(1, 4)`, el valor de `sb` cambia a “po”.
- **insert**: Inserta un elemento en la posición indicada en la cadena. Al ejecutar `sb.insert(2, ‘‘abc’’)` su valor cambia a “peabcdro”.
- **setCharAt**: Modifica el caracter en la posición indicada. Llamar al método `sb.setCharAt(4, ’a’)` modifica `sb` a “pedra”.

3. Ejercicios prácticos

Ver directorio `CodigosT8`.