Control 2

Ejercicios

Profesor: Alexandre Bergel
Auxiliares: Juan-Pablo Silva
Ignacio Slater
Semestre: Primavera 2019

1. Preguntas conceptuales

1. ¿Qué beneficios tiene escribir los tests primero?
2. Nombre las etapas de *TDD* y de una breve descripción de cada uno.
3. ¿Qué es un test unitario? ¿Para qué se usan?
4. ¿Qué diferencia un *framework* de una librería?
5. Si un programa tiene 100 % de *coverage* ¿Significa esto que el programa no tiene errores? Explique.
6. Se tiene la siguiente implementación de un inventario. Cree los tests necesarios para probar el funcionamiento del programa.

```java
import java.util.Map;
import java.util.Map.Entry;
import java.util.TreeMap;

public class Inventario {
  private Map<String, Integer> elementos;

  public Inventario() {
    elementos = new TreeMap<>();
  }

  public void agregarElemento(String nombre, int stock) {
    elementos.put(nombre, stock);
  }

  public boolean contiene(String nombre) {
    return elementos.containsKey(nombre);
  }

  public int obtenerStock(String nombre) {
    return elementos.get(nombre);
  }

  @Override
```

```
  public String toString() {
    StringBuilder resultado = new StringBuilder();
    for (Entry<String, Integer> elemento :
        elementos.entrySet()) {
      resultado.append(elemento.getKey()).append(": ").append(elemento.getValue());
    }
    return resultado.toString();
  }
}
```

7. Usted tiene un computador en el que desea escribir programas en *Python* y *Java*. Para editar los archivos usted tiene 3 *IDEs*: *PyCharm*, *IntelliJ* y *VSCode*. Cada editor puede abrir distintos tipos de archivos, *PyCharm* solamente puede abrir programas escritos en *Python*, *IntelliJ* archivos *Java*, mientras que *VSCode* puede abrir ambos tipos. Una posible implementación de esto se ve a continuación. Comente qué le parece la implementación propuesta.

```
public enum TipoArchivo {
  PYTHON, JAVA
}

public class Archivo {
  private final TipoArchivo tipo;

  public Archivo(TipoArchivo tipo) {
    this.tipo = tipo;
  }

  public TipoArchivo getTipo() {
    return tipo;
  }
}

public class PyCharm {
  public void abrirArchivo(Archivo archivo) {
    if (archivo.getTipo() == TipoArchivo.PYTHON) {
      System.out.println("Archivo abierto con PyCharm");
    } else {
      System.out.println("PyCharm no puede abrir el archivo.");
    }
  }
}

public class IntelliJ {
  public void abrirArchivo(Archivo archivo) {
    if (archivo.getTipo() == TipoArchivo.JAVA) {
      System.out.println("Archivo abierto con IntelliJ");
    } else {
      System.out.println("IntelliJ no puede abrir el archivo.");
    }
  }
}
```

```java
public class VSCode {
  public void abrirArchivo(Archivo archivo) {
    System.out.println("Archivo abierto con VSCode");
  }
}
```

8. ¿Por qué es una mala práctica usar `instanceof`? ¿En qué casos es válido usarlo?
9. ¿Cuáles son las etapas de desarrollo iterativo?
10. De un ejemplo concreto de un problema en el que utilizaría *Adapter Pattern*
11. ¿Para qué se ocupa *Proxy Pattern*?
12. Ilustre mediante diagramas *UML* los patrones *Adapter* y *Proxy*
13. Suponga que tiene una aplicación para realizar cálculos matemáticos en un servidor. Los usuarios de la aplicación acceden desde su navegador e ingresan en formato texto la expresión que quiere calcular. Sin embargo, para que el servidor realice un cálculo, debe recibir un objeto de tipo *MathematicalExpression*. Señale que patrón de diseño utilizaría para resolver este problema y haga un diagrama *UML* para ilustrar su solución.
14. Señale las diferencias entre el *Observer* antiguo de *Java* y el nuevo. ¿Por qué se implementó este cambio?
15. ¿Por qué es mejor utilizar *Observer* para notificar cambios de propiedades en vez de hacer que la clase notifique directamente?
16. ¿En qué casos recomendaría el uso de *Factory Pattern*? ¿En cuáles no?
17. Muestre con un diagrama *UML* un ejemplo de uso de *Factory*.

2. Ejercicios

1. Programe una solución para el problema 7 de la sección anterior utilizando buenas metodologías de diseño.
2. Suponga que quiere modelar una pizzería. La pizzería puede atender clientes de forma presencial y por teléfono. Adicionalmente, su pizzería tiene múltiples hornos para satisfacer los pedidos. Por simplicidad suponga que un sólo empleado se encarga de todo, y debe estar atento en todo momento por si llega un cliente, llaman por teléfono o alguno de los hornos terminó. Proponga una solución para este problema.
3. Considere la misma pizzería de la pregunta anterior. La carta de pizzas que se pueden pedir está definida, por lo que las pizzas a preparar nunca serán alguna que no esté en la lista. Las pizzas que se ofrecen son:
   - **Pollo BBQ**: Queso mozzarella, pollo, tocino, cebolla, salsa BBQ.
   - **Todas las carnes**: Queso mozzarella, pepperoni, jamón, tocino, salchicha italiana.
   - **Vegetariana**: Queso mozzarella, cebolla, pimentón, tomate, aceitunas, champiñón.
   - **Vegana**: Queso vegano, cebolla, pimentón, tomate, aceitunas, champiñón.

   Implemente una manera de resolver este problema.