**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN**

**CCPG1042 - DISEÑO DE SOFTWARE**

**SEGUNDO EXAMEN – II PAO 2023**

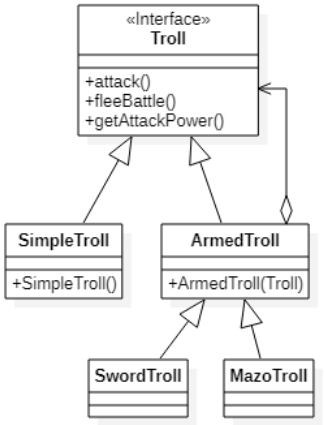
***Nombre:*** _____ ***Matrícula:*** _____

| |
|---|
| 100 |
| Firma |

## Sección A

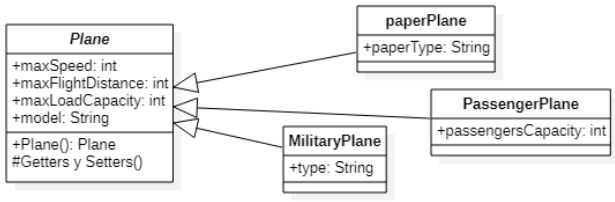
**Justifique su selección de la respuesta, según corresponda. Cada pregunta de esta sección vale 5%.**

1. Identifique el patrón de diseño empleado en el siguiente Diagrama de clases de un personaje de un juego para PC.
    a. Builder
    b. Factory Method
    c. Adapter
    d. Decorator
    e. Observer
    f. Composite




2. Para los siguientes requerimientos de un juego para PC, identifique el patrón de diseño correspondiente:
   *El Rey Orco da las órdenes a su ejército. El ejército está organizado por la siguiente jerarquía [* ***Comandante -> Oficial -> Soldado*** *] y cada uno puede realizar un único tipo de orden. Sin embargo, en caso de no poder realizar la orden asignada se la transfiere a su subordinado.*
    a. Iterator
    b. Chain of responsibility
    c. Decorator
    d. Builder
    e. Factory Method
    f. Strategy

3. En el siguiente diagrama de clases se muestra la relación que existe entre cuatro clases de aviones. Según el código de la clase **PaperPlane** identifique un mal olor de programación (Code smells):




```
1   public class PaperPlane  extends Plane {
2       protected String paperType;
3       public PaperPlane(String type) {
4           super("Paper", 0,0,0);
5           this.paperType = type;
6       }
7       @Override
8       public int getMaxFlightDistance(){
9           throw new Exception(message: "No se puede sensar");
10          return 0;
11      }
12      @Override
13      public int getMaxSpeed(){
14          throw new Exception(message: "No se puede sensar");
15          return 0;
16      }
17      @Override
18      public int getMaxLoadCapacity(){
19          throw new Exception(message: "No se puede sensar");
20          return 0;
21      }
22  }
```

    a. Liskov Sustitution
    b. Feature Envy
    c. Lazy Class
    d. Data Clumps
    e. Refused Bequest
    f. Data Class

4. Indique el ***tipo_assert*** correspondiente para el código de la prueba **testNoMoverEnCruz** para los movimientos de las piezas en un tablero de ajedrez.

```
public class MoveUtil
{
    public static boolean isDiagonalMove(Position from, Position to)
    { …
    }

    public static boolean isHorizontalOrVerticalMove(Position from, Position to)
    { …
    }

    public static boolean isStraightLineMove(Position from, Position to)
    { …
    }
}
```

```
@Test
public void testNoMoverEnCruz()
{
    Position fromPosition = new Position(7, 2);
    Position toPositionDiag1 = new Position(8, 3);
    Position toPositionDiag2 = new Position(6, 3);

    tipo_assert(MoveUtil.isHorizontalOrVerticalMove(fromPosition, toPositionDiag1));
    tipo_assert(MoveUtil.isHorizontalOrVerticalMove(fromPosition, toPositionDiag2));
}
```

   a. assertThrows
   b. assertFalse
   c. assertTrue
   d. assertEquals
   e. assertSame
   f. fail

5. El code smell de "**Comments**" se da cuando:
   a. Los comentarios están mezclados con el código de un método
   b. Hay líneas de código comentadas
   c. No se coloca la documentación de un método
   d. Hay un método difícil de entender
   e. Hay comentarios javadoc para explicar lo que hace un método

6. Indique la técnica de refactorización que utilizaría en el siguiente segmento de código fuente.
   a. Replace Error Code with Exception
   b. Form Template Method
   c. Replace Delegation with Inheritance
   d. Extract method
   e. Replace Parameter with Explicit Methods
   f. Dead code

```
void setValue (String name, int value) {
  if (name.equals("height")) {
   height = value;
   return;
  }
  if (name.equals("width")) {
    width = value;
    return;
  }
  Assert.shouldNeverReachHere();
}
```

## Sección B

7. Considere el escenario que se presenta a continuación:

*El sistema ECommerceHub es una plataforma de comercio electrónico que permite a distintas tiendas en línea vender una amplia variedad de productos a través de una interfaz unificada. Entre las características de este sistema se indican las siguientes:*

*El sistema debe conectarse a una base de datos específica, garantizando que el acceso a esta sea único y centralizado para que cualquier cambio de configuración, plataforma o credenciales sea transparente para el resto del sistema.*

*El sistema debe manejar un producto básico que tiene nombre, descripción y precio. Pero se desea que cada tienda pueda agregar de forma sencilla nuevas características y comportamientos a sus productos en tiempo de ejecución. Inicialmente se ha pensado en características para productos de vestimenta y productos deportivos, siendo una opción para cada tienda elegir las características que deba tener cada uno de sus productos.*

*El sistema toma como parte importante, del servicio que brinda, el correcto manejo de sugerencias y reclamos (llamados incidentes) de los clientes. Por lo tanto, se desea que cada incidente ingresado sea atendido por la tienda dueña del producto, si el cliente aún no queda satisfecho se lo deriva al supervisor de reclamos y si el cliente aún está insatisfecho, el incidente pasa al departamento legal quien tiene la última palabra y decidirá como solucionar dicho incidente.*

a. **Identifique** los patrones de diseño aplicables. **Justifique** su respuesta. **[12%]**
b. Elabore un diseño utilizando **diagrama de clases** que muestre solo la parte relevante a los patrones de diseño seleccionados. De ser posible, identifique herencias, multiplicidades, visibilidad de atributos y métodos. Indique, por medio de notación UML, si las entidades corresponden a interfaces, clases abstractas o clases concretas. Indique cualquier asunción que realice. **[15%]**
c. Separe en **paquetes** según los patrones usados. **[03%]**

## Sección C

8. Considere el siguiente código fuente que corresponde a un sistema de historial médico para un hospital. Los puntos suspensivos indican que el código del método está completo y no tiene ningún code smell adicional.

```java
class PatientRecord {
  private int patientId;
  private String patientName;
  private String diagnosis;
  private ArrayList<String> medications;
  private ArrayList<String> medicalHistory;

  public PatientRecord(int id, String name) {
    patientId = id;
    patientName = name;
    diagnosis = "";
    medications = new ArrayList<>();
    medicalHistory = new ArrayList<>();
  }

  public int getPatientId() {…
  public String getPatientName() {…
  public String getDiagnosis() {…
  public void setDiagnosis(String diag) {…

  public ArrayList<String> getMedications() {
    return medications;
  }

  public void addMedication(String med, String dosis,
      String duration) {
    String timestamp = LocalDateTime.now().toString();
    medications.add(timestamp + ": " + med + " : "
      + dosis + " : " + duration);
  }

  public String getLastMedicalHistory() {
    return medicalHistory.get(medicalHistory.size() - 1);
  }

  public void updateMedicalHistory(String history) {
    String timestamp = LocalDateTime.now().toString();
    medicalHistory.add(timestamp + ": " + history);
  }

  public ArrayList<String> searchMedicalHistory(String keyword) {
    ArrayList<String> matchingHistory = new ArrayList<>();
    TreeMap<Integer, ArrayList<String>> relevanceMap;
    relevanceMap = new TreeMap<>(Collections.reverseOrder());
    String log = "";

    if (medicalHistory.size() > 10) {
      log = "Our patient";
      System.out.println(log);
    }

    for (String history : medicalHistory) {
      int keywordCount = countOccurrences(history, keyword);
      if (keywordCount > 0) {
        ArrayList arr;
        arr = relevanceMap.computeIfAbsent(keywordCount, k -> new ArrayList<>());
        arr.add(history);
      }
    }

    for (Map.Entry<Integer, ArrayList<String>> entry : relevanceMap.entrySet()) {
      matchingHistory.addAll(entry.getValue());
    }

    return matchingHistory;
  }

  private int countOccurrences(String history, String keyword) {
    int count = 0;
    int index = history.indexOf(keyword);
    while (index != -1) {
      count++;
      index = history.indexOf(keyword, index + 1);
    }
    return count;
  }
}
```

A usted se le solicita:

a. Señalar los malos olores de programación en el código e indicar su nombre. Explique por qué sería un problema. **[16%]**

b. Indicar las técnicas de refactorización que utilizaría para mejorar el código. Justifique su respuesta. **[16%]**

**c.** Realizar 2 pruebas unitarias para el método **searchMedicalHistory().** Indique el propósito de cada prueba. **[08%]**