Orientación a Objetos II
2025

Explicación de práctica
Semana del 7 de abril

Cuadernillo Patrones

Orientación a Objetos 2

Cuadernillo Semestral de Actividades

- Patrones de diseño -

Actualizado: 14 de marzo de 2025

El presente cuadernillo **estará en elaboración** durante el semestre y tendrá un compilado con todos los ejercicios que se usarán durante la asignatura. Se irán agregando ejercicios al final del cuadernillo para poder poner en práctica los contenidos que se van viendo en la materia.

Cada semana les indicaremos cuáles son los ejercicios en los que deberían enfocarse para estar al día y algunos de ellos serán discutidos en la explicación de práctica.

Recomendación importante:
Los contenidos de la materia se incorporan y fijan mejor cuando uno intenta aplicarlos - no alcanza con ver un ejercicio resuelto por alguien más. Para sacar el máximo provecho de los ejercicios, es importante asistir a las consultas de práctica habiendo intentado resolverlos (tanto como sea posible). De esa manera, las consultas estarán más enfocadas y el docente podrá dar un mejor feedback.

Ejercicio 1.a: Friday the 13th en Java

La clase Biblioteca implementa la funcionalidad de exportar el listado de sus socios en formato JSON. Para ello define el método **`exportarSocios()`** de la siguiente forma:

```java
/**
 * Retorna la representación JSON de la colección de socios.
 */
public String exportarSocios() {
    return exporter.exportar(socios);
}
```

Ejercicio 1.a: Friday the 13th en Java

La Biblioteca delega la responsabilidad de exportar en una instancia de la clase **VoorheesExporter** que dada una colección de socios, retorna un texto con la representación de la misma en formato JSON. Esto lo hace mediante el mensaje de instancia **`exportar(List<Socio>)`**.

```
1 /**
2  * Retorna la representación JSON de la colección de socios.
3  */
4 public String exportarSocios() {
5     return exporter.exportar(socios);
6 }
```

Ud. puede probar la funcionalidad ejecutando el siguiente código:

```java
public static void main(String[] args) {
    Biblioteca biblioteca = new Biblioteca();
    biblioteca.agregarSocio(new Socio(
            "Arya Stark",
            "needle@stark.com",
            "5234-5"));
    biblioteca.agregarSocio(new Socio(
            "Tyron Lannister",
            "tyron@thelannisters.com",
            "2345-2"));

    System.out.println(biblioteca.exportarSocios());
}
```

Al ejecutar, el mismo imprimirá el siguiente JSON:

```
[
    {
     "nombre": "Arya Stark",
     "email": "needle@stark.com",
     "legajo": "5234-5"
    },
    {
     "nombre": "Tyron Lannister",
     "email": "tyron@thelannisters.com",
     "legajo": "2345-2"
    }
]

```

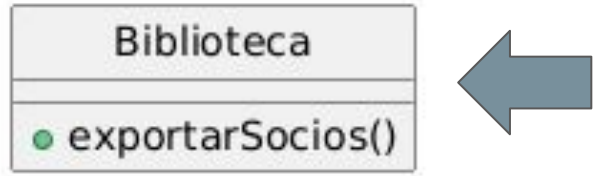
Ejercicio 1.a: Friday the 13th en Java

Tareas:

1. Analice la implementación de la clase Biblioteca, Socio y VoorheesExporter que se provee con el material adicional de esta práctica (Archivo biblioteca.zip).
2. Documente la implementación mediante un diagrama de clases UML.
3. Programe los Test de Unidad para la implementación propuesta.

```java
public class Biblioteca {
    private List<Socio> socios;
    private VoorheesExporter exporter;

    public Biblioteca() {
        socios = new ArrayList<>();
        exporter= new VoorheesExporter();
    }

    public void agregarSocio(Socio socio) {
        socios.add(socio);
    }

    /**
     * Retorna la representación JSON de la colección de socios.
     */
    public String exportarSocios() {
        return exporter.exportar(socios);
    }

    public VoorheesExporter getExporter() {
        return exporter;
    }

    public void setExporter(VoorheesExporter exporter) {
        this.exporter = exporter;
    }
}
```

No se puede modificar!



No se puede modificar!

```java
public class Socio {
    private String nombre;
    private String legajo;
    private String email;

    public Socio(String nombre, String email, String legajo) {
        this.nombre = nombre;
        this.email = email;
        this.legajo = legajo;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    // Getters y Setters
}
```

```java
public class VoorheesExporter {

    private String exportar(Socio socio) {
        String separator = System.lineSeparator();
        return "\t{" + separator
            + "\t\t\"nombre\": \"" + socio.getNombre() + "\"," + separator
            + "\t\t\"email\": \"" + socio.getEmail() + "\"," + separator
            + "\t\t\"legajo\": \"" + socio.getLegajo() + "\"" + separator
            + "\t}";
    }

    public String exportar(List<Socio> socios) {
        if (socios.isEmpty()) {
            return "[]";
        }
        String separator = System.lineSeparator();
        StringBuilder buffer = new StringBuilder("[" + separator);
        socios.forEach(socio → {
            buffer.append(this.exportar(socio))
                .append(",")
                .append(separator);
        });
        // remueve la última coma y fin de línea
        buffer.setLength(buffer.length() - (separator.length() + 1));
        buffer.append(separator).append("]");
        return buffer.toString();
    }
}
```

No se puede modificar!

Ejercicio 1.b - Usando la librería JSON.simple

La librería **JSON.simple** es liviana y muy utilizada para leer y escribir archivos JSON. Entre las clases que contiene se encuentran:

- **JSONObject**: Usada para representar los datos que se desean exportar de un objeto.
- **JSONArray**: Usada para generar listas.
- **JSONParser**: Usada para recuperar desde un String con formato JSON los elementos que lo componen.

Ejercicio 1.b - Usando la librería JSON.simple

Su nuevo desafío consiste en utilizar la librería JSON.simple para imprimir en formato JSON a los socios de la Biblioteca en lugar de utilizar la clase VoorheesExporter.

Pero con la siguiente condición: **nada de esto debe generar un cambio en el código de la clase Biblioteca.**

Ejercicio 1.b - Usando la librería JSON.simple

1. Instale la librería JSON.simple agregando la siguiente dependencia al archivo **pom.xml** de Maven

```
<dependency>

<groupId>com.googlecode.json-simple</groupId>
    <artifactId>json-simple</artifactId>
    <version>1.1.1</version>
</dependency>
```

Copiar y pegar

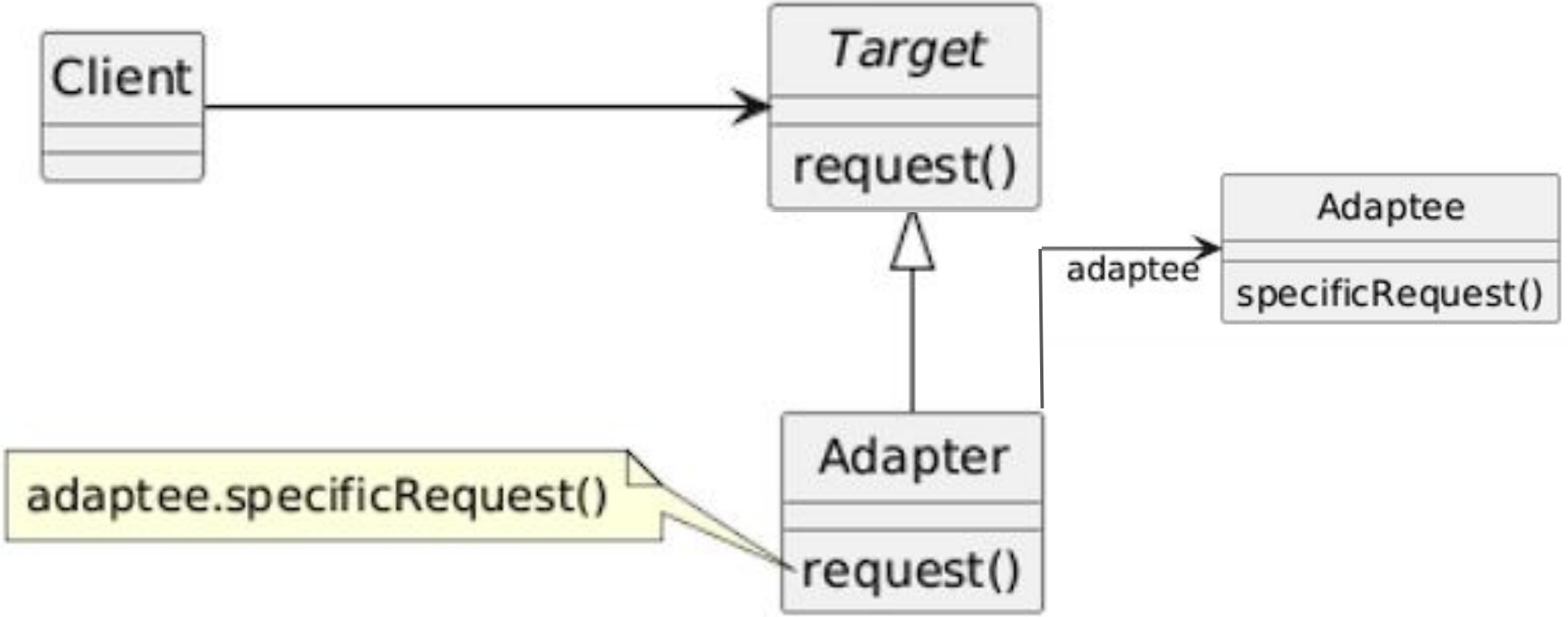
pom.xml

```
1  <dependencies>
2    <dependency>
3        <groupId>org.junit.jupiter</groupId>
4        <artifactId>junit-jupiter-engine</artifactId>
5        <version>5.7.2</version>
6        <scope>test</scope>
7
8
9    </dependency>
10 </dependencies>
```

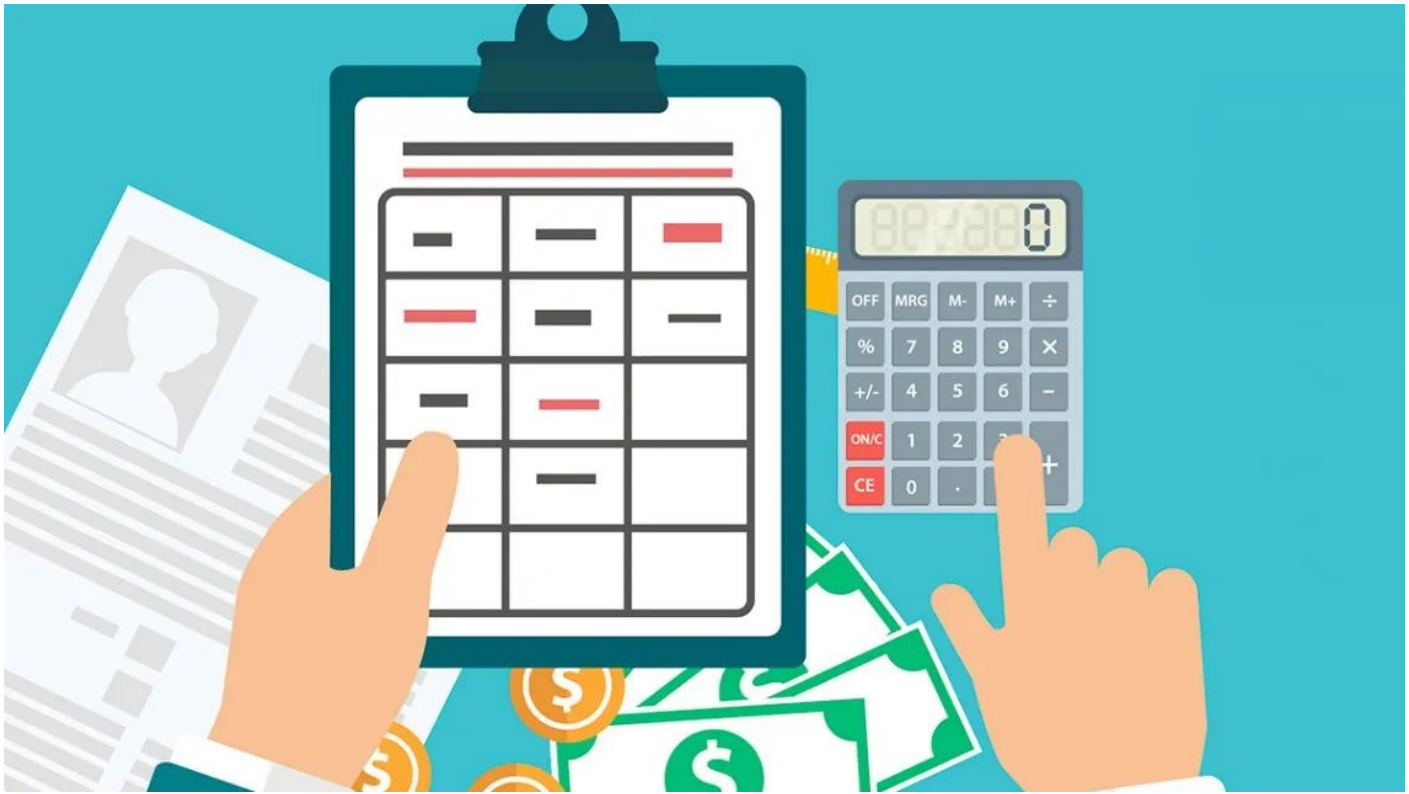
Ejercicio 1.b - Usando la librería JSON.simple

2. Utilice esta librería para imprimir, en formato JSON, los socios de la Biblioteca en lugar de utilizar la clase VoorheesExporter, **sin que esto genere un cambio en el código de la clase Biblioteca**.
   a. Modele una solución a esta alternativa utilizando un diagrama de clases UML. Si utiliza patrones de diseño indique los roles en las clases utilizando estereotipos.
   b. Implemente en Java la solución incluyendo los tests que crea necesarios.

Adapter - Estructura



Ejercicio 2: Cálculo de sueldos



Ejercicio 2: Cálculo de sueldos

	Temporario	Pasante	Planta
básico	$ 20.000 + cantidad de horas que trabajó * $ 300.	$20.000	$ 50.000
adicional	$5.000 si está casado $2.000 por cada hijo	$2.000 por examen que rindió	$5.000 si está casado $2.000 por cada hijo $2.000 por cada año de antigüedad
descuento	13% del sueldo básico 5% del sueldo adicional	13% del sueldo básico 5% del sueldo adicional	13% del sueldo básico 5% del sueldo adicional

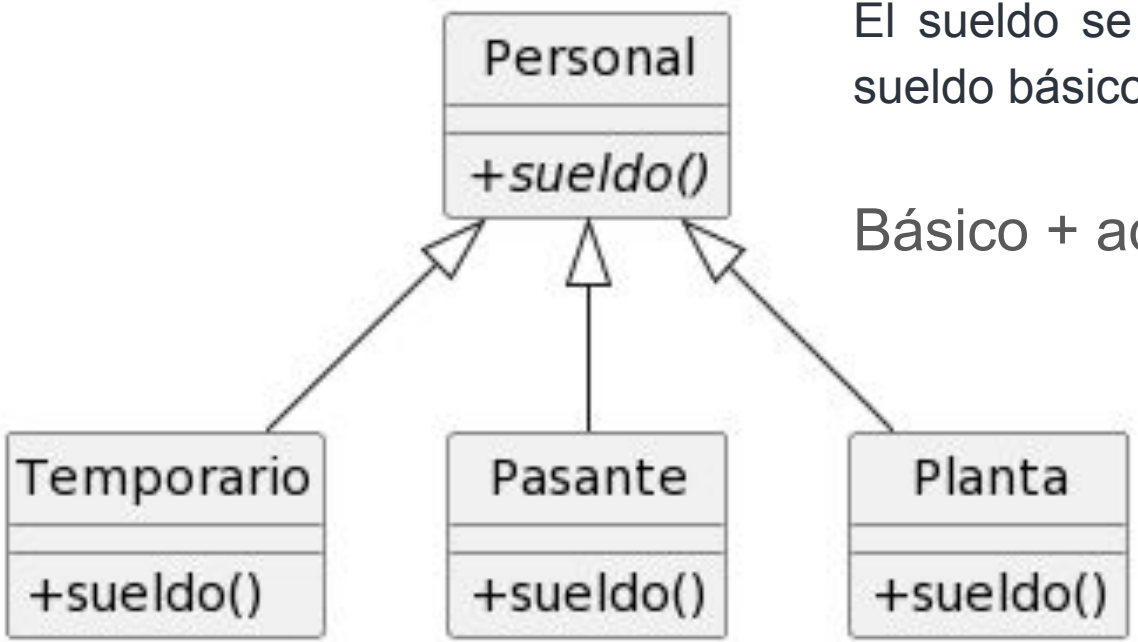
Ejercicio 2: Cálculo de sueldos

Sea una empresa que paga sueldos a sus empleados, los cuales están organizados en tres tipos: Temporarios, Pasantes y Planta. El sueldo se compone de 3 elementos: sueldo básico, adicionales y descuentos.

Tareas:

1. Diseñe la jerarquía de Empleados de forma tal que cualquier empleado puede responder al mensaje **#sueldo.**
2. Desarrolle los test cases necesarios para probar todos los casos posibles.
3. Implemente en Java.

Diseño



El sueldo se compone de 3 elementos: sueldo básico, adicionales y descuentos.

Básico + adicional - descuento

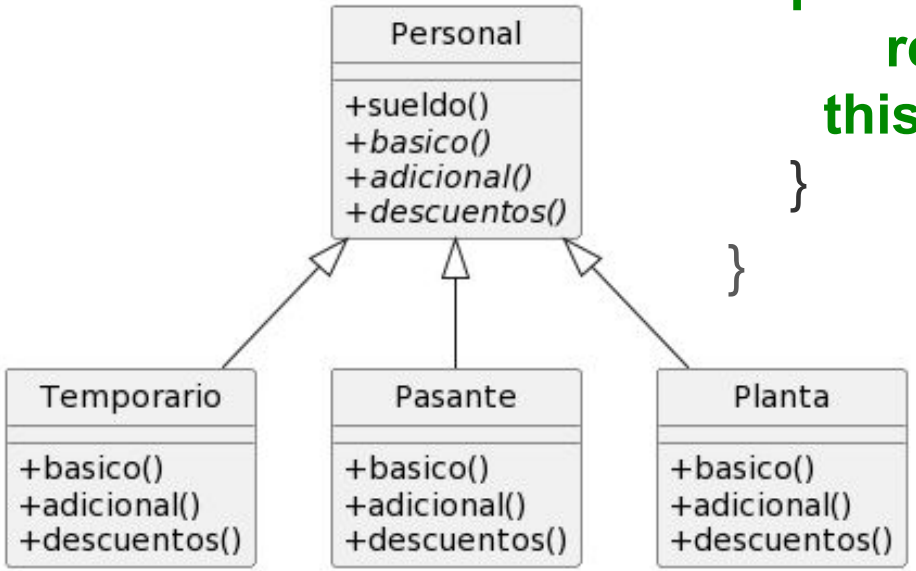
Diseño

El sueldo se compone de 3 elementos: sueldo básico, adicionales y descuentos.

Básico + adicional - descuento

```
public class Personal {
    public int sueldo() {
        return this.basico()+ this.adicional()- this.descuento();
    }
}
```

Diseño




```
public class Personal {
    public int sueldo() {
        return this.basico()+
      this.adicional()- this.descuento();
    }
}
```

Cálculos

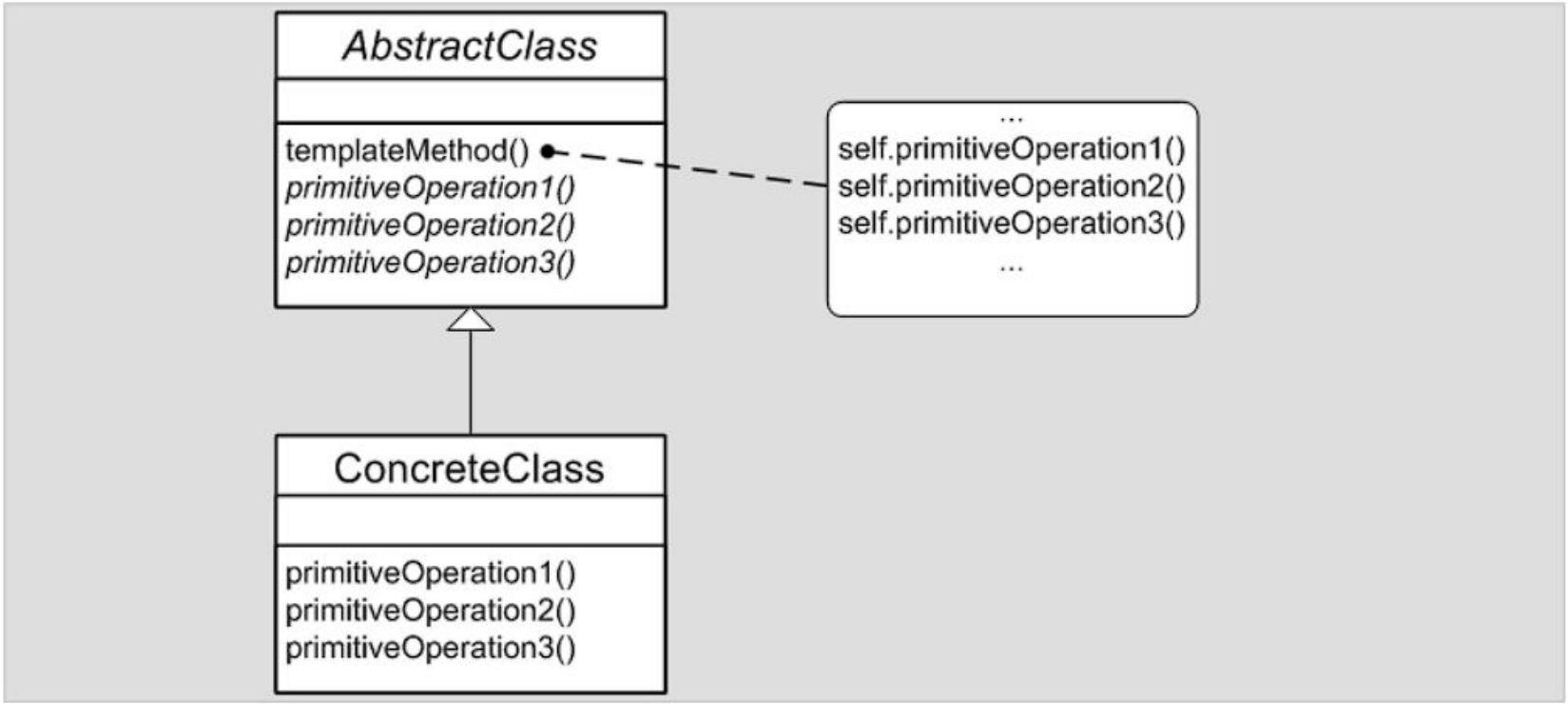
```
public class Temporario{
  public int basico(){
    return ???
  }

  public int adicional(){
    return ???
  }

  public int descuentos(){
    return ???
  }
```

```
public class Pasante{
  public int basico(){
      return ???
  }

  public int adicional(){
      return ???
  }

      public int descuentos(){
      return ???
  }
```

```
public class Planta{
  public int basico() {
      return ???
  }

  public int adicional(){
      return ???
  }

  public int descuentos(){
      return ???
  }
```

¿Qué pasa con descuentos?

Template Method - Estructura



Ejercicio 3: Media Player

Usted ha implementado una clase Media player, para reproducir archivos de audio y video en formatos que usted ha diseñado. Cada Media se puede reproducir con el mensaje play().

Para continuar con el desarrollo, usted desea incorporar la posibilidad de reproducir Video Stream.

Para ello, dispone de la clase VideoStream que pertenece a una librería de terceros y usted no puede ni debe modificarla. El desafío que se le presenta es hacer que la clase MediaPlayer pueda interactuar con la clase VideoStream.

Ejercicio 3: Media Player