1. Escriba un ejemplo del uso de herencia, detallando para cada clase cuál es su rol dentro de la herencia y los motivos por los cuales cree que es correcta la implementación. El ejemplo debe ser original, se ruega evitar el uso de casos comunes como Animal o Vehículo.

2. Explique el concepto de herencia múltiple y cuál es el problema que se presenta al implementarla. ¿Cuál es la solución que brinda Java a este problema?

3. Ejemplifique el uso de interfaces y explique si es mejor tener interfaces mas genericas o mas especificas.

4. Defina la clase abstracta Personaje con los atributos vida, nivelAtaque, nivelDefensa y los métodos atacar():Integer y defender(Integer puntos). Implemente el método atacar pero no el método defender(). Luego, cree dos clases hijas donde ambas implementan el método defenderse y al menos una de ellas extiende el método atacar. Cada clase debe establecer una cantidad de puntos de vida por defecto.

   En un ataque se deben realizar cierta cantidad de puntos de daño y en la defensa se debe reducir esa cantidad de puntos de daños. El resultado final de los puntos de ataque debe descontar la misma cantidad de puntos de vida del personaje que defiende.

5. Usando las clases del ejercicio anterior, realizar tres combates entre instancias de ellas. En un combate cada personaje realiza un ataque por turno y es perdedor aquel que queda primero sin vida. En caso de que la vida llegue a un valor negativo el sistema debe lanzar una excepción y en el control de la misma debe dejarse la vida en 0. La decisión de cual personaje ataca primero debe ser aleatoria.

6. Observe y analice el siguiente código:
   Sin ejecutar el código, ¿puede deducir qué se va a imprimir **exactamente** en el foreach de la clase Main? ¿Por qué?.

Persona:

```
public abstract class Persona {
        private String nombre,apellido;
        public Persona(String nombre, String apellido) {
                super();
                this.nombre = nombre;
                this.apellido = apellido;
```

```
	}

	public String getNombre() {
		return nombre;
	}

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

	public String getApellido() {
		return apellido;
	}

	public void setApellido(String apellido) {
		this.apellido = apellido;
	}

	public abstract String materia();

}
```

Alumno:

```
import java.util.ArrayList;
public class Alumno extends Persona{

	ArrayList<String> materias = new ArrayList<>();

	public Alumno(String nombre, String apellido) {
		super(nombre, apellido);
	}

	public ArrayList<String> getMaterias() {
		return materias;
	}

	public void setMaterias(ArrayList<String> materias) {
		this.materias = materias;
	}

	public void agregarMateria(String materia) {
		this.materias.add(materia);
	}

	public String materia() {
		String msg = "El Alumno se encuentra cursando las siguientes materias \n";
		for (String materia : materias) {
			msg = msg + materia + "\n";
		}
		return msg;
	}
```

```
}
```

Docente:

```
import java.util.ArrayList;
public class Docente extends Persona{

	private String materia;

	public Docente(String nombre, String apellido, String materia) {
		super(nombre, apellido);
		this.materia = materia;
	}

	public String getMateria() {
		return materia;
	}

	public void setMateria(String materia) {
		this.materia = materia;
	}

	public String materia() {
		return "El docente dicta la siguiente clase " + this.materia +"\n";
	}

}
```

Main:

```
import java.util.ArrayList;
import java.util.Random;
public class Main {

	public static void main(String[] args) {
		Random rnd = new Random();
		ArrayList<Persona> personas = new ArrayList<>();
		for(int i = 0; i < 10; i++) {
			if(rnd.nextBoolean()) {
				personas.add(new Docente("NomDoc"+i, "ApeDoc"+i, "Mat"+i));
			}else {
				Alumno alumno = new Alumno("AlumNom"+i, "AlumApe"+i);
				for(int j = 0; j < 10; j++) {
					alumno.agregarMateria("Mat"+j);
					if(rnd.nextBoolean()) {
						break;
					}
				}
				personas.add(alumno);
			}
		}
```

```
                    for (Persona persona : personas) {
                        System.out.println(persona.materia());
                    }

                }

            }
```

7. Para el código del ejercicio anterior indique cómo haría usted para implementar polimorfismo estático.

8. En pos de favorecer la reutilización de código, reescriba el siguiente utilizando herencia.

Archivo ProfesorTitular.java

```
package eje8;
public class ProfesorTitular {
        private String nombre;
        private String apellido;
        private Integer edad;
        private Integer horasTrabajadas;
        private Integer antiguedad;
        private final Double valorAntiguedad = 1000.0;
        private final Double valorHora = 300.0;

        public ProfesorTitular(String nombre, String apellido, Integer edad, Integer horasTrabajadas,
Integer antiguedad){
            this.setNombre(nombre);
            this.setApellido(apellido);
            this.setEdad(edad);
            this.setHorasTrabajadas(horasTrabajadas);
            this.setAntiguedad(antiguedad);
        }

        /* ATENCIÓN: Aquí deben ir getters && setter para  Nombre, Apellido, Edad,
HorasTrabajadas,Antiguedad */

        public Double getRemuneracionPorAntiguedad() {
            return this.valorAntiguedad * this.getAntiguedad();
        }
        public Double getRemuneracionMensual() {
            return (this.valorHora * this.getHorasTrabajadas()) +
this.getRemuneracionPorAntiguedad();
        }
}
```

Archivo ProfesorSuplente.java

```
package eje8;
public class ProfesorSuplente {
        private String nombre;
```

```
	private String apellido;
	private Integer edad;
	private Integer horasTrabajadas;
	private final Double valorHora = 200.0;

	public ProfesorSuplente(String nombre, String apellido, Integer edad, Integer horasTrabajadas){
		this.setNombre(nombre);
		this.setApellido(apellido);
		this.setEdad(edad);
		this.setHorasTrabajadas(horasTrabajadas);
	}

	/* ATENCIÓN: Aquí deben ir getters && setter para  Nombre, Apellido, Edad,
HorasTrabajadas,Antiguedad */

	public Double getRemuneracionMensual() {
		return this.valorHora * this.getHorasTrabajadas();
	}
}
```

Archivo Main.java

```
package eje8;
import java.util.*;

public class Main {

	public static void main(String[] args) {

		ArrayList<ProfesorTitular> titulares = new ArrayList<ProfesorTitular>();
		titulares.add(new ProfesorTitular("Aldana", "Gutierrez", 40, 40, 15));
		titulares.add(new ProfesorTitular("Pedro", "Perez", 30, 30, 4));
		titulares.add(new ProfesorTitular("María", "Gomez", 29, 40, 1));

		ArrayList<ProfesorSuplente> suplentes = new ArrayList<ProfesorSuplente>();
		suplentes.add(new ProfesorSuplente("Tomas", "Sosa", 28, 40));
		suplentes.add(new ProfesorSuplente("Luciana", "Torres", 35, 10));


		for (int i = 0; i < titulares.size(); i++) {
			System.out.println("Nombre y apellido: " + titulares.get(i).getNombre() + " " +
titulares.get(i).getApellido());
			System.out.println("¿Es titular?: Sí");
			System.out.println("Remuneración:" +
titulares.get(i).getRemuneracionMensual());
		}

		for (int i = 0; i < suplentes.size(); i++) {
			System.out.println("Nombre y apellido: " + suplentes.get(i).getNombre() + " " +
suplentes.get(i).getApellido());
			System.out.println("¿Es titular?: No");
			System.out.println("Remuneración:" +
suplentes.get(i).getRemuneracionMensual());
		}
```

```
        }
    }
```

9. Defina la clase TarifaProveedor con un método calcular(totalSMS, totalMinutos, totalGigas) que, dado la cantidad de mensajes, minutos de llamada y GB de consumo de datos calcule el valor en pesos a pagar. El valor a retornar del método calcular debe ser la suma de los resultados obtenidos en los métodos calcularSMS(totalSMS), calcularMinutosDeLlamada(totalMinutos) y calcularConsumoGB(totalGigas)

   Los valores por defecto de cada servicio son
   - Mensajes de texto(SMS): $1
   - Minuto de llamada: $15
   - Gigas(GB) de internet: $20.

   Además de los métodos anteriores, debe poseer un método abstracto getNombre() que retorne el nombre del proveedor.

   Luego, defina una clase hija por cada uno de los siguientes proveedores:
   - Claro: que tiene un 20% extra sobre el básico total
   - Personal: que tiene un 20% extra sobre los minutos de llamada y 50% sobre los GB de datos.
   - Movistar: tiene un 10% extra sobre los mensajes de texto, 20% sobre las llamadas y 30% sobre los GB de datos.

   Desarrolle una interfaz gráfica que permita ingresar la cantidad de SMS, minutos de llamada, Gigas y muestre como resultado la tarifa que se obtendría con cada proveedor.

10. Desarrollé parte del sistema de control de horarios para la universidad.

    Para todo el personal que trabaja en la universidad se desea tener el nombre completo, antigüedad en años y sector. Además, es necesario que para cada uno sea posible almacenar y obtener la cantidad de horas trabajadas en el mes.

    El personal docente tiene asociada una categoría que define la cantidad de horas semanales a trabajar:

- Simple: 10 hs semanales
- Semiexclusiva: 20 hs semanales
- Exclusiva: 40 hs semanales

El personal no docente puede tener dos tipos de jornadas asignadas:

- Completa: 30 hs semanales
- Reducida: 20 hs semanales

Defina la clase Reloj que sepa generar un informe. Este debe indicar para cada persona la cantidad de horas trabajadas en el mes indicando además si cumplió o no con el mínimo esperado.

Para poner a prueba el sistema debe realizarse un simulación con los siguientes criterios:

- Al menos diez personas entre docentes y no docentes
- Para los docentes debe definirse aleatoriamente la categoría
- Para los no docentes debe definirse aleatoriamente la jornada
- La cantidad de horas trabajadas en el mes debe ser generada aleatoriamente bajo las siguientes pautas:
    - Para docentes con categoría simple hay un 95% de posibilidad de que exceda la cantidad de horas requeridas.
    - Para docentes con categoría semiexclusiva hay un 75% de posibilidad de que exceda la cantidad de horas requeridas.
    - Para docentes con categoría exclusiva hay un 60% de posibilidad de que exceda la cantidad de horas requeridas.
    - Para no docentes hay un 80% de posibilidad de que exceda la cantidad de horas requeridas.
- Debe solicitarse a una instancia de la clase Reloj que imprima el informe.