Trabajo Práctico
Patrón Proxy

1. El siguiente programa consta de una clase Persona y una clase Teléfono, donde una persona puede tener uno o varios teléfonos. La clase PersonaDao que permite obtener de una base de datos una instancia de Persona dado su identificador, y un Main que obtiene una persona e imprime su nombre y los teléfonos que posee. Para poder ejecutar este programa, en su base de datos preferida, genere las siguientes tablas (que permitirán modelar a personas y su relación uno a muchos con teléfonos):

```
personas (id: int, nombre: varchar(100));
telefonos (id: int, numero: varchar(20), idPersona: int); idPersona es foraña de personas.
```

Además, implemente el método privado *PersonasDao#obtenerConexion*() a su gusto. Finalmente, inserte una persona con varios teléfonos y ejecute el método *Main#main* para comprobar su funcionamiento.

```
public class Main {

	public static void main(String args[]) {
		PersonaDao dao = new PersonaDao();
		Persona p = dao.personaPorId(1);
		System.out.println(p.nombre());
		for (Telefono telefono : p.telefonos()) {
			System.out.println(telefono);
		}
	}
}

public class Telefono {

	private String numero;
	public Telefono(String numero) {
		this.numero = numero;
	}

	public String numero() {
		return numero;
	}
	@Override
	public String toString() {
		return numero;
```

```
		}
}

public class Persona {
		private int id;
		private String nombre;
		private Set<Telefono> telefonos;

		public Persona(int id, String nombre, Set<Telefono> telefonos) {
			this.id = id;
			this.nombre = nombre;
			this.telefonos = telefonos;
		}

		public Telefono[] telefonos() {
			return telefonos.toArray(new Telefono[telefonos.size()]);
		}

		public String nombre() {
			return nombre;
		}
}

public class PersonaDao {

		private Connection obtenerConexion() {
			//Utilice aquí su motor de BD preferido
		}

		public Persona personaPorId(int id) {
			String sql = "select p.nombre,t.numero "
					+ "from personas p, telefonos t "
					+ "where p.id = t.idpersona and p.id = ?";
			try (Connection conn = obtenerConexion();
					PreparedStatement statement =
conn.prepareStatement(sql);) {

				statement.setInt(1, id);
				ResultSet result = statement.executeQuery();
				Set<Telefono> telefonos = new HashSet<Telefono>();
				String nombrePersona = null;
				while (result.next()) {
					nombrePersona = result.getString(1);
					telefonos.add(new Telefono(result.getString(2)));
				}

				return new Persona(id, nombrePersona, telefonos);
```

```
                } catch(SQLException e) {
                        throw new RuntimeException(e);
                }
        }
}
```

Como puede observar, el método *PersonaDao#personaPorId* realiza una consulta SQL para obtener la persona y todos sus teléfonos, y devuelve una instancia de Persona, con su colección de teléfonos.

Usted advierte que hay otros clientes (además de *Main#main*) de *PersonaDao* que no necesitan tener la colección de teléfonos, porque no invocan el método *Persona#telefonos*. Con lo cual podría evitarse realizar una consulta SQL de junta (join) entre dos tablas para la mayoría de los casos.

Utilice el patrón proxy y modifique el método *PersonaDao#personaPorId* de modo tal que la colección de teléfonos en *Persona* se popule únicamente si se invoca al método *Persona#telefonos*. Las clases *Main*, *Persona* y *Telefono* no deben modificarse.

Indique que clases del código entregado son el Cliente, el Proxy y el SujetoReal. **Ayuda**: *L*a interfaz *Set* definida en *Persona* corresponde al Sujeto del patrón Proxy.

2. La empresa Tres Estrellas desea modificar su sistema para agregar control de acceso a los archivos que maneja. El sistema posee, entre otras, las siguientes clases:

```
public enum Permiso {
  ADMIN, BASICO, INTERMEDIO
}

public class Usuario {

  private String name;
  private List<Permiso> permisos;

  public Usuario(String name, List<Permiso> permisos) {
    this.name = name;
    this.permisos = permisos;
  }

  public boolean poseePermiso(Permiso permiso) {
    return permisos.stream().anyMatch(p -> p.equals(permiso));
  }
}


public class FileAccess {
```

```
  private String ruta;
  private String nombreArchivo;

  public FileAccess(String ruta, String nombre) {
    this.ruta = ruta;
    this.nombreArchivo = nombre;
  }

  public String readFile() throws IOException {
    return Files.readString(Paths.get(this.ruta + "/" + this.nombreArchivo));
  }
}
```

Utilizando el patrón Proxy implemente el control de acceso a la lectura de los archivos. Aquellos archivos cuyo nombre comienza con la letra “i” (de importante), solo los usuario con permiso ADMIN pueden accederlos. Los archivos que comienzan con la letra “m”, lo pueden ver los usuarios con permiso ADMIN e INTERMEDIO. Cualquier otro archivo, lo ven todos los usuarios sin importar qué permiso tengan.

Utilice Usuarios#possePermiso para verificar permisos. En caso de intento de lectura sin permiso lance una excepción indicando el error.

Escriba un Main mostrando como se usa.

Realice el diagrama de clases, ponga claramente los método más importantes.