1) ¿Qué principios se violan y por qué?
Supongamos que el método a continuación pertenece a una clase RPNCalculator:

```
public double calculator(String formula) throws OperandException {
        String[] ops = formula.split("\\s+");
        Stack<Double> list = new Stack<Double>();
        double arg2 = 0;

        for (String op: ops){
            switch (op){
            case "+":
                list.push(list.pop()+list.pop());
                break;
            case "-":
                arg2 = list.pop();
                list.push(list.pop() - arg2);
                break;
            case "*":
                list.push(list.pop() * list.pop());
                break;
            case "/":
                arg2 = list.pop();
                list.push(list.pop() / arg2);
                break;
            case "^":
                double power = list.pop();
                list.push(Math.pow(list.pop(), power));
                break;
            default:
                int number = Integer.parseInt(op);
                if ((number < 0) || (number > 99)){
                    throw new OperandException("Not an integer within bounds");
                }
                list.push((double) number);
                break;
```

```
            }
        }
```

Estamos interesados en incorporar nuevas operaciones como: sin, cos, mod

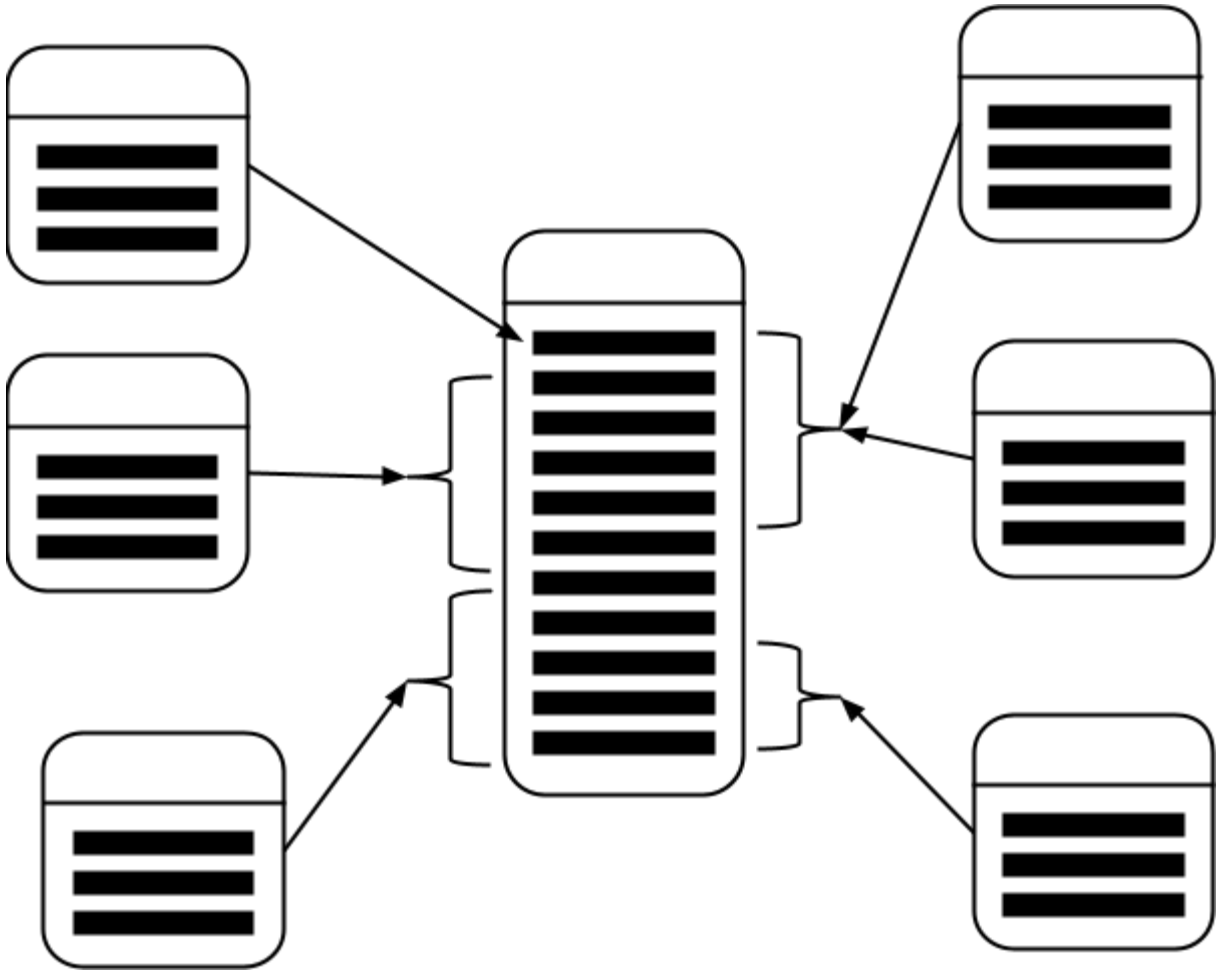
2) ¿Qué principio violan las clases de la JDK involucradas en el siguiente extracto?

```
Vector<String> vectorStack = new Stack<String>();
vectorStack.addElement("one");
vectorStack.addElement("two");
vectorStack.addElement("three");
vectorStack.removeElementAt(1);
System.out.println(vectorStack.size());
// prints: 2
```

3) ¿Qué principios viola este método de la clase Collection de la JDK? ¿Cómo afecta la respuesta saber que lo mismo ocurre en múltiples métodos de la misma?

```
/**
 * Removes all of the elements from this collection (optional operation).
 * The collection will be empty after this method returns.
 *
 * @throws UnsupportedOperationException if the clear operation
 * is not supported by this collection
 */
void clear();
```

4) ¿Qué principios se violan y por qué?



5) ¿Qué principios se violan y por qué?

```
public class ChessGame {
    private boolean atomicChessRule = false;
    private boolean circeChessRule = false;
    private boolean kamikazeChessRule = false;
    private boolean mockChessRule = false;
    // more things ....
    public ChessGame() {
        XMLConfiguration configuration = XMLConfiguration.loadFromDefaultFile();
```

```
        atomicChessRule = configuration.booleanValueOf("atomicChessRule");
        circeChessRule = configuration.booleanValueOf("circeChessRule");
        kamikazeChessRule = configuration.booleanValueOf("kamikazeChessRule");
        mockChessRule = configuration.booleanValueOf("mockChessRule");
        //... and more...
    }
    // more things ....
}
```

6) ¿Se viola algún principio SOLID? En caso afirmativo indicar cuál/es, y cómo lo resolvería.

```
public class XYZPlayer {
    private List<Album> albums;
    //...
    public long countRunningTime() {
        long count = 0;
        for (Album album : albums) {
            for (Track track : album.getTrackList()) {
                count += track.getLength();
            }
        }
        return count;
    }
    public long countMusicians() {
        long count = 0;
        for (Album album : albums) {
            count += album.getMusicianList().size();
        }
        return count;
    }
    public long countTracks() {
        long count = 0;
        for (Album album : albums) {
            count += album.getTrackList().size();
```

```
        }
        return count;
    }
    //...
}
```

7) ¿Se viola algún principio SOLID? En caso afirmativo indicar cuál/es, y cómo lo resolvería.

```
public class MileageCalculator {
    private List<Car> cars;
    public MileageCalculator(List<Car> cars) {
        this.cars = cars;
    }
    public void CalculateMileage() {
        for (Car car : cars) {
            if (car.name.equals("Audi")) {
                System.out.println("Mileage of the car " + car.name + " is
10M");
            } else if (car.name.equals("Mercedes")) {
                System.out.println("Mileage of the car" + car.name + "is 20M");
            }
        }
    }
}
```

8) ¿Se viola algún principio SOLID? En caso afirmativo indicar cuál/es, y cómo lo resolvería. ¿Qué sucede si queremos luego enviar notificaciones por SMS?

```
public class UserManager {
    private EmailNotifier notifier = new EmailNotifier();
    public void CreateUser(String userid, String password, String email) {
        //create user here
        notifier.send(email, "User created successfully!");
    }
    public void ChangePassword(String userid, String oldpassword, String
```

```
newpassword, String email) {
        //change password here
        notifier.send(email, "Password changed successfully");
    }
}
```

9) ¿El siguiente diagrama viola algún principio SOLID? En caso afirmativo indicar cuál/es, y cómo lo resolvería.