Q2

Beverages.java | Tea.java | Coffee.java | BeverageTest.java

```java
package beverages; // Adjust the package name accordingly

abstract class Beverages {  no usages  2 inheritors
    private boolean wantsExtras;  2 usages

    public final void finalTemplateMethod() {...}

    public void setWantsExtras(boolean wantsExtras) {  no usages
        this.wantsExtras = wantsExtras;
    }

    public void boilWater() {  1 usage
        System.out.println("Boiling water");
    }

    public void pourInCup() {  1 usage
        System.out.println("Pouring into cup");
    }

    abstract void brew();  1 usage  2 implementations

    abstract void addCondiments();  1 usage  2 implementations

    abstract void addExtras();  1 usage  2 implementations
}
```

Beverages.java | Tea.java | Coffee.java | BeverageTest.java

```java
package beverages;

class Tea extends Beverages {  2 usages

    @Override  1 usage
    void brew() {
        System.out.println("Steeping the tea");
    }

    @Override  1 usage
    void addCondiments() {
        System.out.println("Adding lemon");
    }

    @Override  1 usage
    void addExtras() {
        System.out.println("Adding honey");
    }
}
```

Beverages.java | Tea.java | Coffee.java | BeverageTest.java

```java
package beverages;

class Coffee extends Beverages {  2 usages

    @Override  1 usage
    void brew() {
        System.out.println("Dripping coffee through filter");
    }

    @Override  1 usage
    void addCondiments() {
        System.out.println("Adding sugar and milk");
    }

    @Override  1 usage
    void addExtras() {
        System.out.println("Adding vanilla syrup");
    }
}
```

Beverages.java | Tea.java | Coffee.java | BeverageTest.java

```java
package beverages;

import java.util.Scanner;

public class BeverageTest {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        Tea tea = new Tea();
        Coffee coffee = new Coffee();

        System.out.print("Do you want extras with your tea (yes/no)? ");
        String teaExtras = scanner.nextLine().trim().toLowerCase();
        tea.setWantsExtras(teaExtras.equals("yes"));

        System.out.print("Do you want extras with your coffee (yes/no)? ");
        String coffeeExtras = scanner.nextLine().trim().toLowerCase();
        coffee.setWantsExtras(coffeeExtras.equals("yes"));

        System.out.println("\nMaking tea...");
        tea.finalTemplateMethod();

        System.out.println("\nMaking coffee...");
        coffee.finalTemplateMethod();

        scanner.close();
    }
}
```

Q3

Shape.java | Square.java × | Circle.java | Rectangle.java | Triangle.java | ShapeFactory.java | ShapeTest.java

```java
package shapes;

public class Square implements Shape {  1 usage
    @Override  4 usages
    public void draw() {
        System.out.println("Square:");
        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++) {
                System.out.print("* ");
            }
            System.out.println();
        }
    }
}
```

Shape.java | Square.java | Circle.java × | Rectangle.java | Triangle.java | ShapeFactory.java | ShapeTest.java

```java
package shapes;

public class Circle implements Shape {  1 usage
    @Override  4 usages
    public void draw() {
        System.out.println("Circle:");
        int radius = 6;  // Adjust radius to match the desired size

        // Iterate over each row
        for (int i = 0; i <= 2 * radius; i++) {
            // Iterate over each column
            for (int j = 0; j <= 2 * radius; j++) {
                // Calculate the distance from the center of the circle
                double distance = Math.sqrt((i - radius) * (i - radius) + (j - radius) * (j - radius));

                // If the distance is less than or equal to the radius, print a star
                if (distance > radius - 0.5 && distance < radius + 0.5) {
                    System.out.print("* ");
                } else {
                    System.out.print("  ");
                }
            }
            System.out.println();
        }
    }
}
```

Shape.java | Square.java | Circle.java | Rectangle.java | Triangle.java | ShapeFactory.java | ShapeTest.java

```java
package shapes;

public class Rectangle implements Shape {  1 usage
    @Override  4 usages
    public void draw() {
        System.out.println("Rectangle:");
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 6; j++) {
                System.out.print("* ");
            }
            System.out.println();
        }
    }
}
```

Shape.java | Square.java | Circle.java | Rectangle.java | Triangle.java | ShapeFactory.java | ShapeTest.java

```java
package shapes;

public class Triangle implements Shape {  1 usage
    @Override  4 usages
    public void draw() {
        System.out.println("Triangle:");
        int rows = 5; // Example number of rows for triangle
        for (int i = 1; i <= rows; i++) {
            for (int j = rows; j > i; j--) {
                System.out.print(" ");
            }
            for (int k = 1; k <= (2 * i - 1); k++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

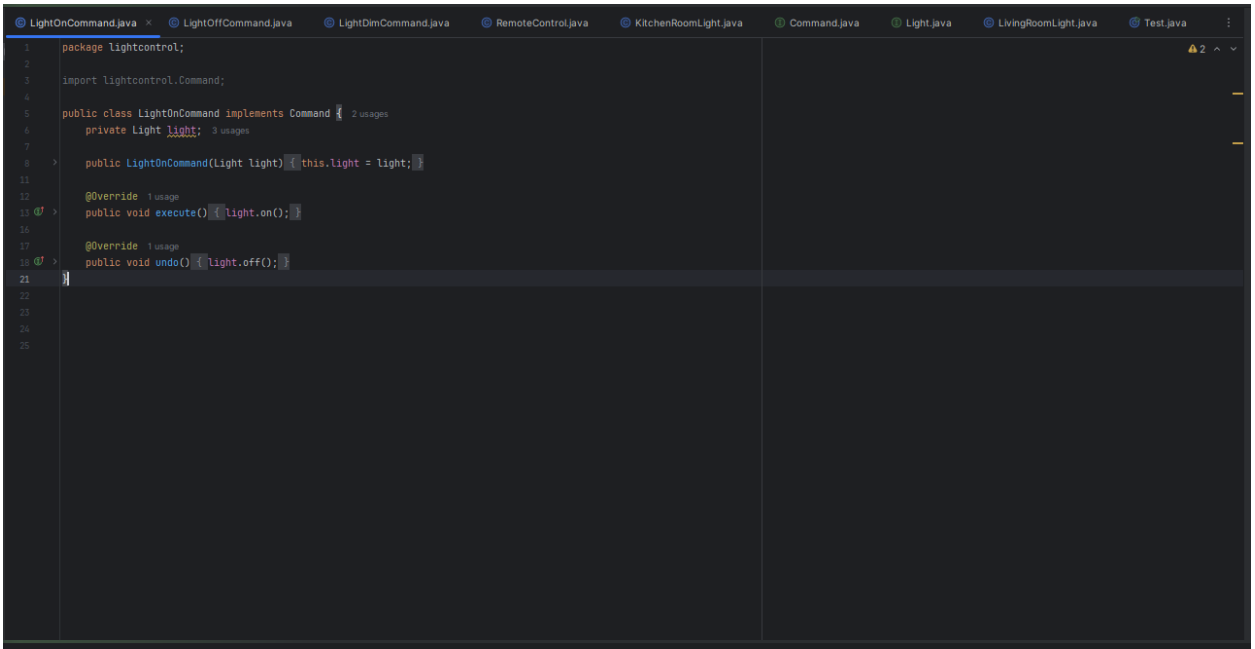
Shape.java | Square.java | Circle.java | Rectangle.java | Triangle.java | ShapeFactory.java | ShapeTest.java

```java
package shapes;

public class ShapeFactory {  2 usages
    // Use getShape method to get the object of the shape type
    public Shape getShape(String shapeType) {  4 usages
        if (shapeType == null) {
            return null;
        }
        if (shapeType.equalsIgnoreCase( anotherString: "SQUARE")) {
            return new Square();
        } else if (shapeType.equalsIgnoreCase( anotherString: "CIRCLE")) {
            return new Circle();
        } else if (shapeType.equalsIgnoreCase( anotherString: "RECTANGLE")) {
            return new Rectangle();
        } else if (shapeType.equalsIgnoreCase( anotherString: "TRIANGLE")) {
            return new shapes.Triangle();
        } else {
            System.out.println("Invalid shape type provided.");
            return null;
        }
    }
}
```

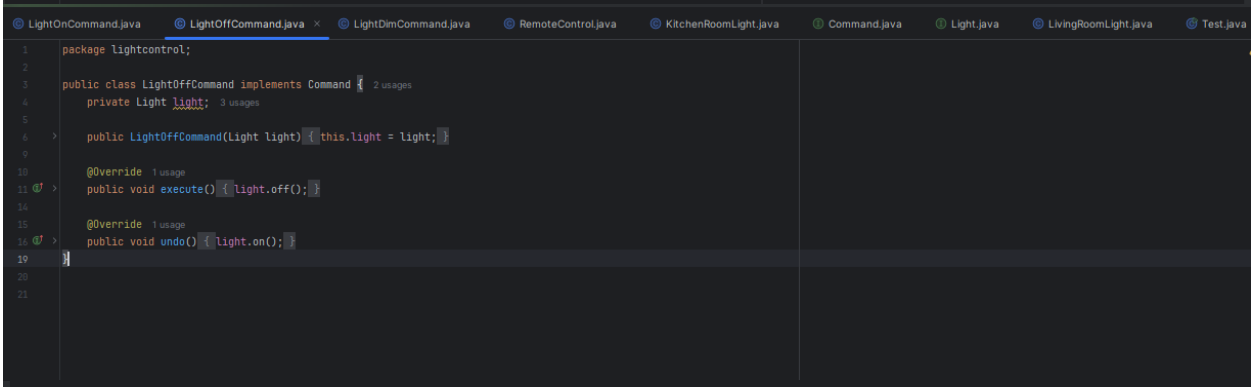
Shape.java | Square.java | Circle.java | Rectangle.java | Triangle.java | ShapeFactory.java | ShapeTest.java

```java
package shapes;

public class ShapeTest {
    public static void main(String[] args) {
        ShapeFactory shapeFactory = new ShapeFactory();

        // Create and draw a Square
        Shape shape1 = shapeFactory.getShape( shapeType: "SQUARE");
        if (shape1 != null) shape1.draw();

        // Create and draw a Circle
        Shape shape2 = shapeFactory.getShape( shapeType: "CIRCLE");
        if (shape2 != null) shape2.draw();

        // Create and draw a Rectangle
        Shape shape3 = shapeFactory.getShape( shapeType: "RECTANGLE");
        if (shape3 != null) shape3.draw();

        // Create a shape that is invalid
        Shape shape4 = shapeFactory.getShape( shapeType: "INVALID_SHAPE");
        if (shape4 != null) shape4.draw();
    }
}
```

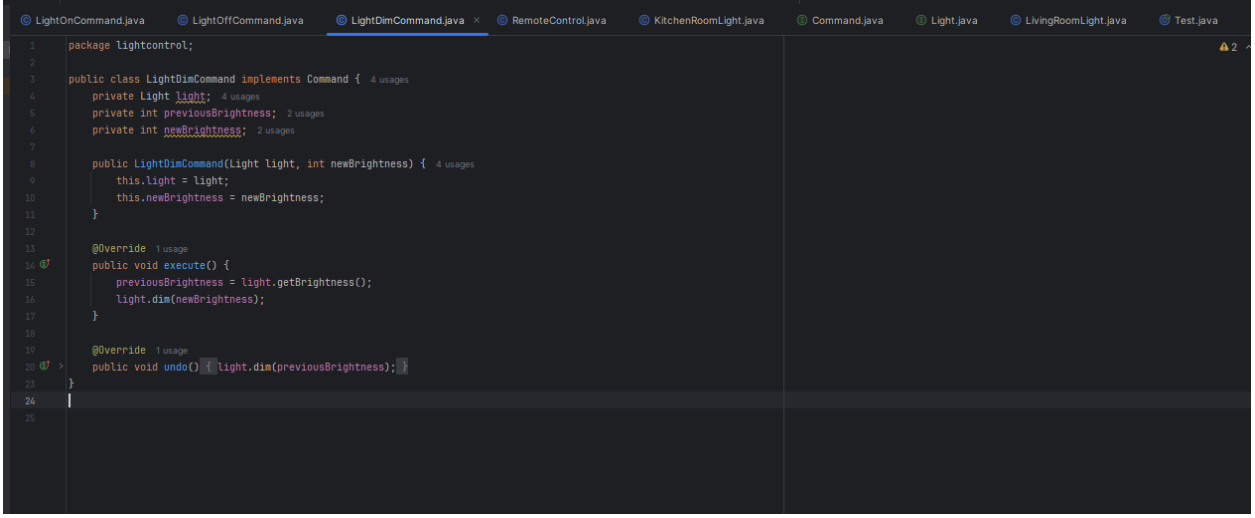
Q1



```java
package lightcontrol;

import lightcontrol.Command;

public class LightOnCommand implements Command {
    private Light light;

    public LightOnCommand(Light light) { this.light = light; }

    @Override
    public void execute() { light.on(); }

    @Override
    public void undo() { light.off(); }
}
```



```java
package lightcontrol;

public class LightOffCommand implements Command {
    private Light light;

    public LightOffCommand(Light light) { this.light = light; }

    @Override
    public void execute() { light.off(); }

    @Override
    public void undo() { light.on(); }
}
```



```java
package lightcontrol;

public class LightDimCommand implements Command {
    private Light light;
    private int previousBrightness;
    private int newBrightness;

    public LightDimCommand(Light light, int newBrightness) {
        this.light = light;
        this.newBrightness = newBrightness;
    }

    @Override
    public void execute() {
        previousBrightness = light.getBrightness();
        light.dim(newBrightness);
    }

    @Override
    public void undo() { light.dim(previousBrightness); }
}
```

LightOnCommand.java | LightOffCommand.java | LightDimCommand.java | RemoteControl.java | KitchenRoomLight.java | Command.java | Light.java | LivingRoomLight.java | Test.java

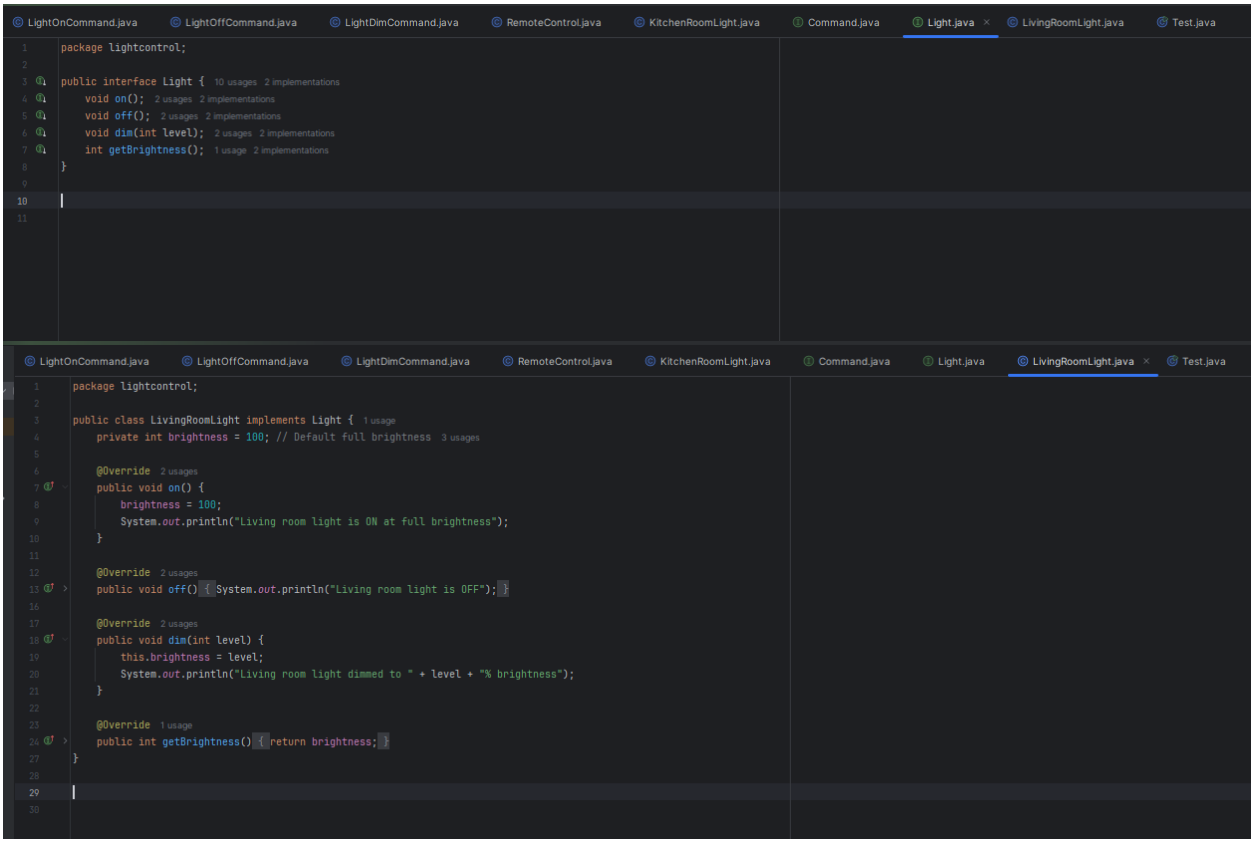
```java
package lightcontrol;

public class RemoteControl {  2 usages
    private Command[] commands;  5 usages
    private Command lastCommand;  3 usages

    public RemoteControl(int numSlots) { commands = new Command[numSlots]; }

    public void setCommand(int slot, Command command) { commands[slot] = command; }

    public void pressButton(int slot) {  10 usages
        if (commands[slot] != null) {
            commands[slot].execute();
            lastCommand = commands[slot];
        }
    }

    public void pressUndo() {  no usages
        if (lastCommand != null) {
            lastCommand.undo();
        } else {
            System.out.println("No command to undo");
        }
    }
}
```

LightOnCommand.java | LightOffCommand.java | LightDimCommand.java | RemoteControl.java | KitchenRoomLight.java | Command.java | Light.java | LivingRoomLight.java | Test.java

```java
package lightcontrol;

public class KitchenRoomLight implements Light {  1 usage
    private int brightness = 100; // Default full brightness  3 usages

    @Override  2 usages
    public void on() {
        brightness = 100;
        System.out.println("Kitchen room light is ON at full brightness");
    }

    @Override  2 usages
    public void off() { System.out.println("Kitchen room light is OFF"); }

    @Override  2 usages
    public void dim(int level) {
        this.brightness = level;
        System.out.println("Kitchen room light dimmed to " + level + "% brightness");
    }

    @Override  1 usage
    public int getBrightness() { return brightness; }
}
```

LightOnCommand.java | LightOffCommand.java | LightDimCommand.java | RemoteControl.java | KitchenRoomLight.java | Command.java | Light.java | LivingRoomLight.java | Test.java

```java
package lightcontrol;

public interface Command {  16 usages  3 implementations
    void execute();  1 usage  3 implementations
    void undo();  1 usage  3 implementations
}
```


LightOnCommand.java
LightOffCommand.java
LightDimCommand.java
RemoteControl.java
KitchenRoomLight.java
Command.java
Light.java
LivingRoomLight.java
Test.java
package lightcontrol;
public interface Light { 10 usages 2 implementations
void on(); 2 usages 2 implementations
void off(); 2 usages 2 implementations
void dim(int level); 2 usages 2 implementations
int getBrightness(); 1 usage 2 implementations
}
LightOnCommand.java
LightOffCommand.java
LightDimCommand.java
RemoteControl.java
KitchenRoomLight.java
Command.java
Light.java
LivingRoomLight.java
Test.java
package lightcontrol;
public class LivingRoomLight implements Light { 1 usage
private int brightness = 100; // Default full brightness 3 usages
@Override 2 usages
public void on() {
brightness = 100;
System.out.println("Living room light is ON at full brightness");
}
@Override 2 usages
public void off() { System.out.println("Living room light is OFF"); }
@Override 2 usages
public void dim(int level) {
this.brightness = level;
System.out.println("Living room light dimmed to " + level + "% brightness");
}
@Override 1 usage
public int getBrightness() { return brightness; }
}