Student 1: Shaman Ranjan (2023498)
Student 2: Siddharth Yadav (2023525)

Angry Birds Deadline-3

Overview

This project is a simplified version of the popular **Angry Birds** game, implemented using the **LibGDX** framework. The game involves launching birds to destroy structures and pigs placed within those structures. The game is divided into multiple levels, each with its unique layout of birds, blocks, and pigs.

Project Structure

The project is organized into several packages and classes, each responsible for different aspects of the game:

Packages

- **com.angryBirds.Birds**:
  Contains classes for different types of birds: `RedBird`, `BlueBird`, `YellowBird`.

- **com.angryBirds.Blocks**:
  Contains classes for different types of blocks: `Cube`, `Plank`, `Column`, `Triangle`, `Wall`, `Ground`.

- **com.angryBirds.Levels**:
  Contains classes for different levels: `Level_1`, `Level_2`, `Level_3` and the base level class `BaseLevel`.

- **com.angryBirds.Pigs**:
  Contains classes for different types of pigs: `NormalPig`, `KingPig`.

- **com.angryBirds.Utils**:
  Contains utility classes: `SaveData`, `musicControl`.

- **com.angryBirds.Screens**:
  Contains classes for different game screens.

Key Classes

Main

The main class initializes the game.

BaseLevel

An abstract class providing common functionality for all levels, such as:

- Loading textures.
- Setting up the launcher.
- Handling game objects.

Levels

- **Level_1, Level_2, Level_3**:
  Concrete implementations of different levels, each with a unique layout of birds, blocks, and pigs.

Launcher

Handles the mechanism for launching birds.

CollisionHandler

Implements the `ContactListener` interface to handle collisions between game objects.

Block

An abstract class representing a block in the game. Subclasses include:

- `Cube`
- `Plank`
- `Column`
- `Triangle`
- `Wall`
- `Ground`

Bird

An abstract class representing a bird in the game. Subclasses include:

- `RedBird`
- `BlueBird`

- `YellowBird`

Pig

An abstract class representing a pig in the game. Subclasses include:

- `NormalPig`
- `KingPig`

Implemented Design Patterns

Template Method

```
public abstract class BaseLevel implements Screen {

   protected abstract void initializeGameObjects();

   // Template methods shared across levels

   protected void addBird(Bird bird)

   protected void addBlock(Block block)

   protected void addPig(Pig pig)

}
```

- BaseLevel is an abstract class that defines the skeleton of game levels
- Specific levels like Level_1 and Level_3 override initializeGameObjects()
- Common functionality is implemented in the base class

Factory Method Pattern

```
private void loadGameObjects(SaveData saveData) {
   Bird bird = createBird(birdData);
   Block block = createBlock(blockData);
   Pig pig = createPig(pigData);
}
```

- Factory methods for creating game objects like birds, blocks and pigs

Observer Pattern

```
protected void checkEndCondition() {
    boolean allPigsDead = true;
    boolean allBirdsInactive = true;
    // Observer monitors game state changes
    if (allPigsDead) {
        game.setScreen(new WinScreen(game));
    } else if (allBirdsInactive && !allPigsDead) {
        game.setScreen(new LossScreen(game));
    }
}
```

- Game state is observed to detect win/loss conditions

Singleton Pattern

```
private musicControl mc;
mc = musicControl.getInstance();
```

- Music control uses a singleton pattern to ensure only one instance

---

OOPs concepts used

Inheritance

```
// Base class for all levels
public abstract class BaseLevel implements Screen {
    // Common level functionality
}

// Level implementations inheriting from BaseLevel
public class Level_1 extends BaseLevel {
    // Specific Level 1 implementation
}

public class Level_3 extends BaseLevel {
    // Specific Level 3 implementation
```

```
}
```

Abstraction

```
// Abstract base class defining level structure
public abstract class BaseLevel implements Screen {
    protected abstract void initializeGameObjects();
}

// Abstract methods force implementation in child classes
@Override
protected void initializeGameObjects() {
    addBird(new RedBird(game, 100, ground_height, world));
    addBlock(new Ground(game, "stone", 0, 0, world));
    // etc
}
```

Encapsulation

```
// Private fields with protected/public accessors
protected Main game;
protected World world;
private musicControl mc;

// Protected methods for internal use
protected void addBird(Bird bird)
protected void addBlock(Block block)
protected void addPig(Pig pig)
```

Polymorphism

```
// Interface implementation
public abstract class BaseLevel implements Screen {
    public void render(float delta)
    public void dispose()
}

// Method overriding in subclasses
@Override
public void render(float delta) {
    super.render(delta);
```

```
    // Custom rendering
}
```

Composition

```
java
public abstract class BaseLevel {
    protected Stage stage;
    protected World world;
    protected Array<Image> birds;
    protected Array<Image> blocks;
    protected Array<Image> pigs;
}
```

Interface Implementation

```
/// Implementing libGDX Screen interface
public abstract class BaseLevel implements Screen {
    @Override
    public void show() {}

    @Override
    public void hide() {}
}
```

Testing Support

```
private class TestPig extends Pig {
    public TestPig(Main game, float width, float height, World world, float x, float y) {
        super(game, width, height, world, x, y);
    }
}
```

Running this project

Navigate to the project directory

cd AngryBirds

Build the project

./gradlew build

Run the project

./gradlew lwjgl3:run

Run tests

./gradlew test