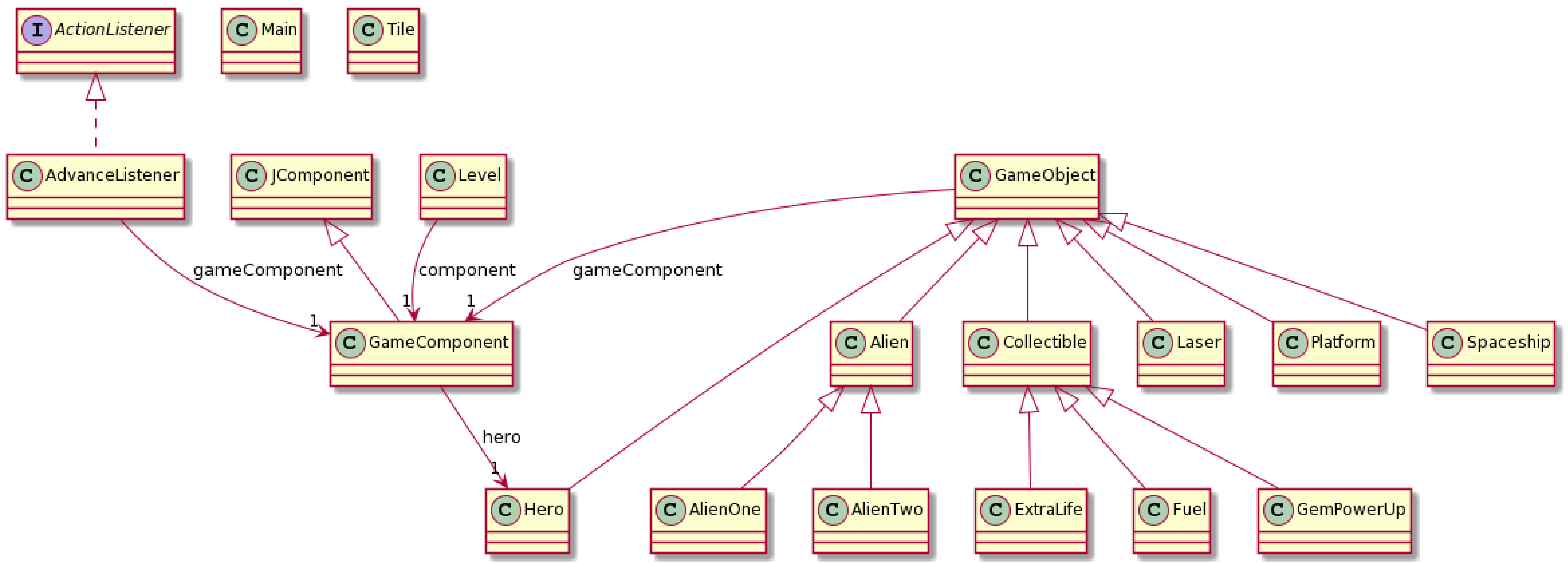
Arcade Game Project

Endia Clark and Annelise Kezdy

Overall Design

- For our overall design, we tried to make sure we broke up pieces of the game into separate classes so that none of our classes would get too big.
- We ended up with fairly high cohesion, but we had a lot of dependencies between classes, so our coupling was also a little bit high.

Plant UML Diagram

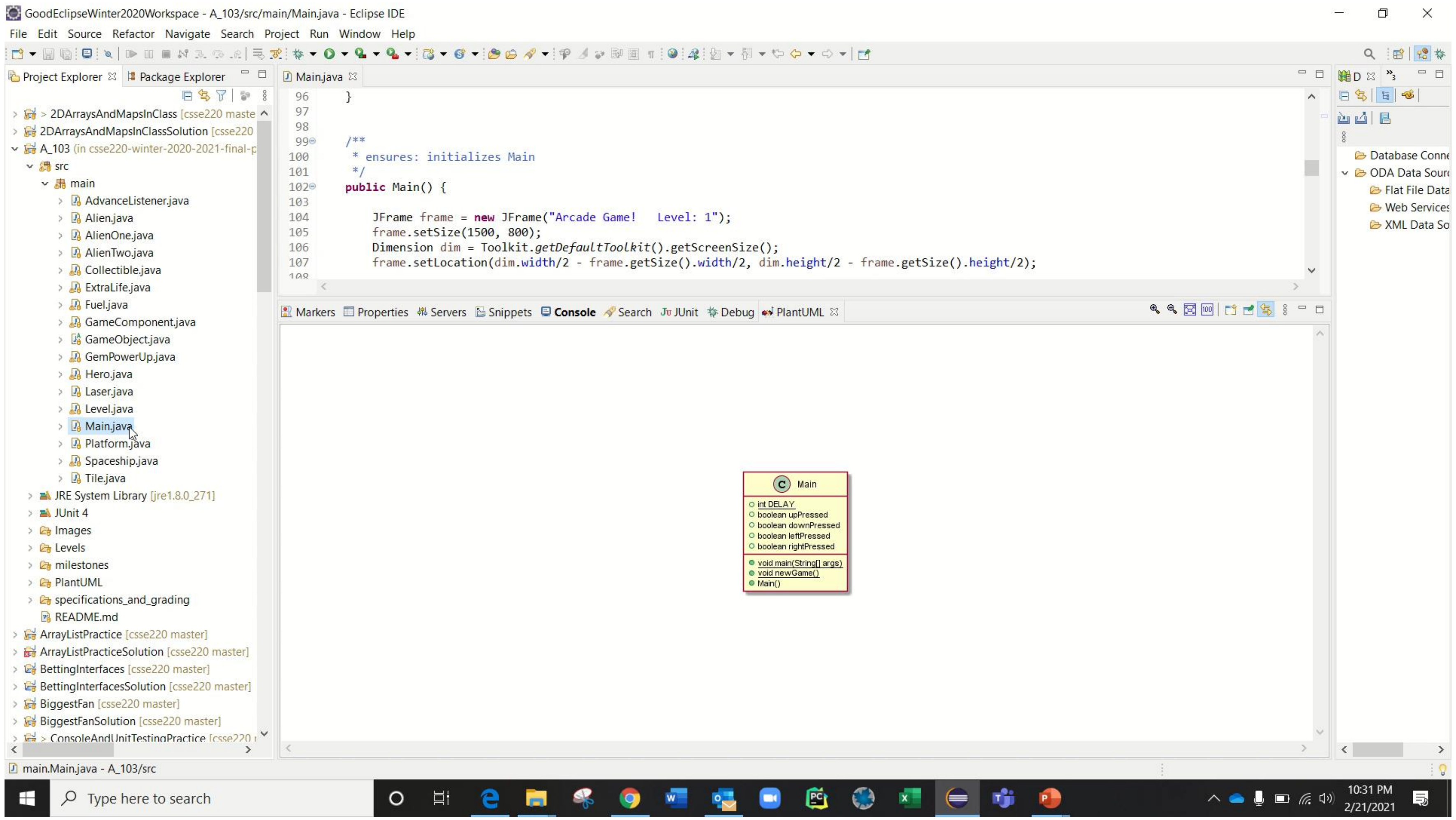


OO Design Principles

- Principle 1: Our design allows for proper functionality.
- Principle 2: Our design was structured around making classes out of different game objects, and each class implements functions specific to itself.
- Principle 3: Though we tried to ensure each class had only one responsibility, our GameComponent class eventually became bigger than we intended as we expanded it to handle most of the main game functions.
- Principle 4: We minimized message chains, but in some instances we still had a lot of dependencies between classes.
- Principle 5: There was some duplicated code in our method of collision handling, but overall we made good use of abstract classes and super classes.

Extra Features

- We implemented a teleportation function, included start and end screens that allowed the game to be restarted without exiting the program, and included special power ups for the hero.


GoodEclipseWinter2020Workspace - A_103/src/main/Main.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Package Explorer
Main.java
Markers Properties Servers Snippets Console Search JUnit Debug PlantUML
Database Conne
ODA Data Sourc
Flat File Data
Web Services
XML Data So
main.Main.java - A_103/src
Type here to search
10:31 PM
2/21/2021

```
> 2DArraysAndMapsInClass [csse220 maste
2DArraysAndMapsInClassSolution [csse220
A_103 (in csse220-winter-2020-2021-final-p
  src
    main
      AdvanceListener.java
      Alien.java
      AlienOne.java
      AlienTwo.java
      Collectible.java
      ExtraLife.java
      Fuel.java
      GameComponent.java
      GameObject.java
      GemPowerUp.java
      Hero.java
      Laser.java
      Level.java
      Main.java
      Platform.java
      Spaceship.java
      Tile.java
  JRE System Library [jre1.8.0_271]
  JUnit 4
  Images
  Levels
  milestones
  PlantUML
  specifications_and_grading
  README.md
ArrayListPractice [csse220 master]
ArrayListPracticeSolution [csse220 master]
BettingInterfaces [csse220 master]
BettingInterfacesSolution [csse220 master]
BiggestFan [csse220 master]
BiggestFanSolution [csse220 master]
> ConsoleAndUnitTestingPractice [csse220
```

```java
    }


    /**
     * ensures: initializes Main
     */
    public Main() {

        JFrame frame = new JFrame("Arcade Game!   Level: 1");
        frame.setSize(1500, 800);
        Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
        frame.setLocation(dim.width/2 - frame.getSize().width/2, dim.height/2 - frame.getSize().height/2);
```

```
Main
int DELAY
boolean upPressed
boolean downPressed
boolean leftPressed
boolean rightPressed
void main(String[] args)
void newGame()
Main()
```

One Major Design Decision

- One major design decision was implementing collision handling. Rather than create an interface, we decided to give the abstract class GameObject, which was the superclass for most objects in the game, abstract methods that each object would implement.