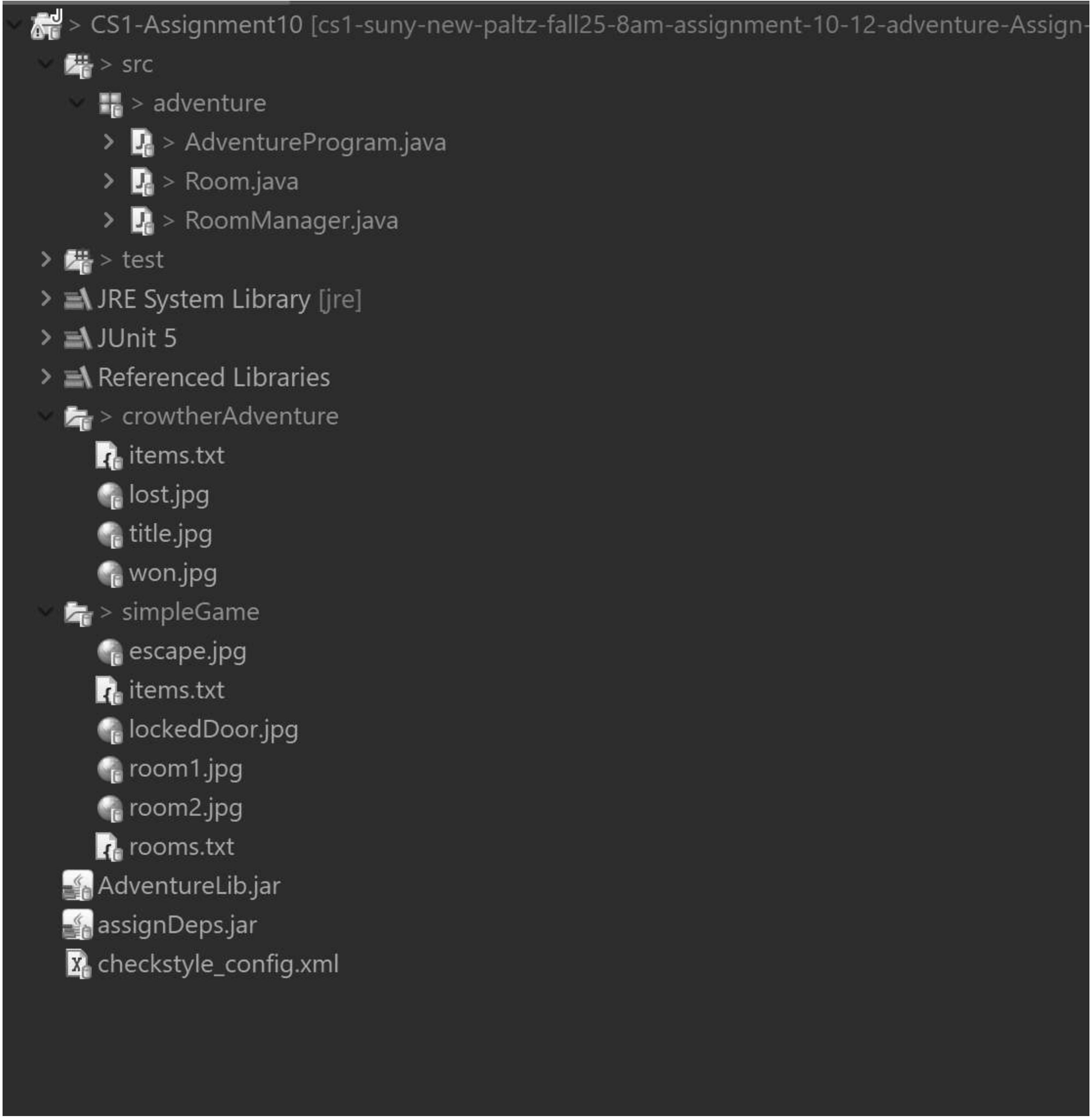
CS1 Adventure Game — Extra Credit assignment Walkthrough

Table of Contents

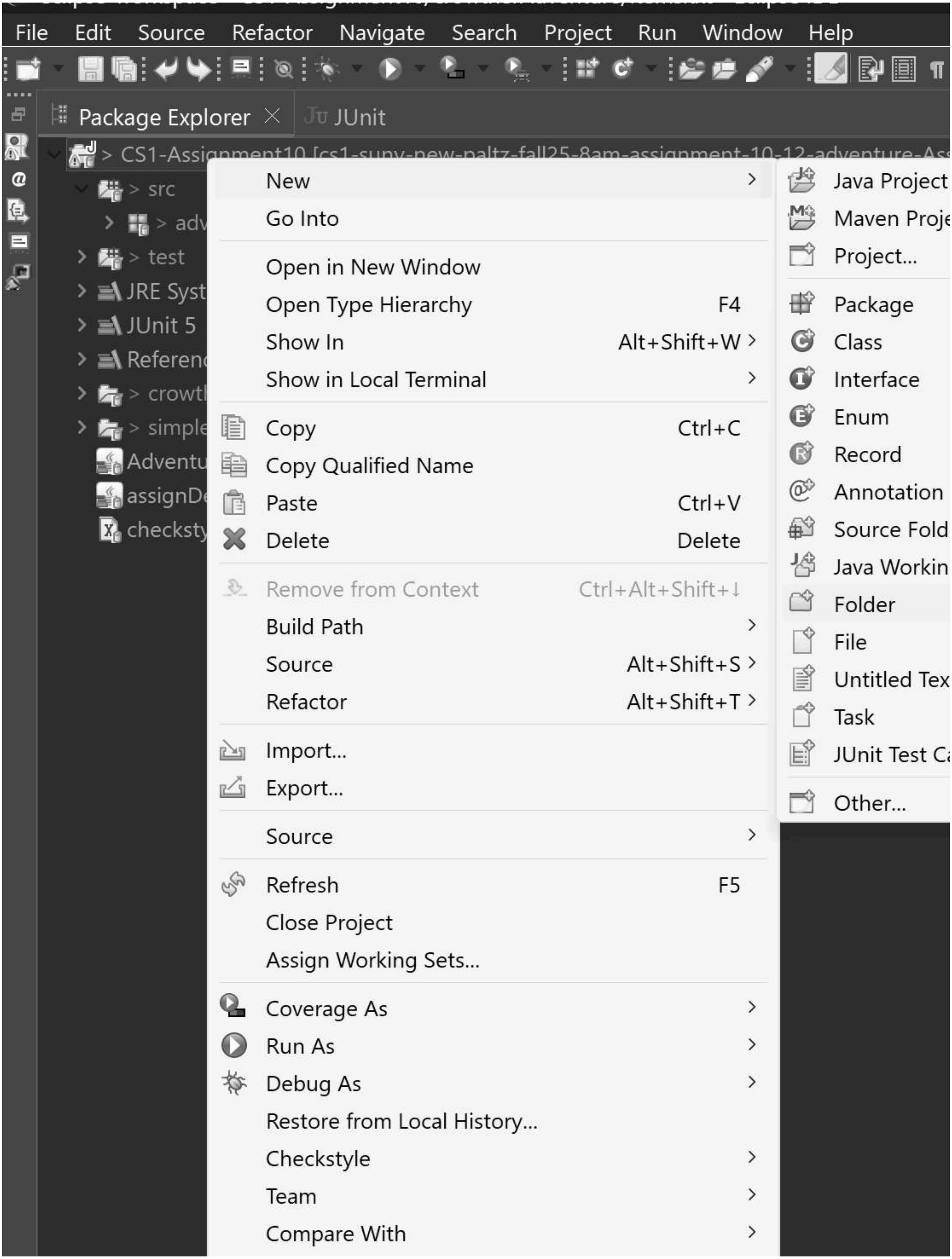
1. Correct Project Structure (This Is Critical)

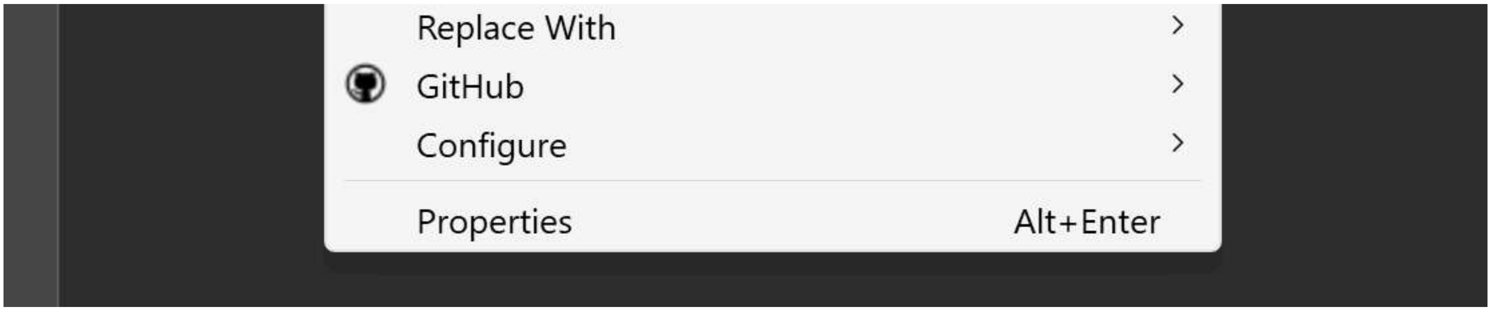
First, open up the CS1-Assignment10-12 project in Eclipse, the one that we have been working on for awhile. You shuld see this:



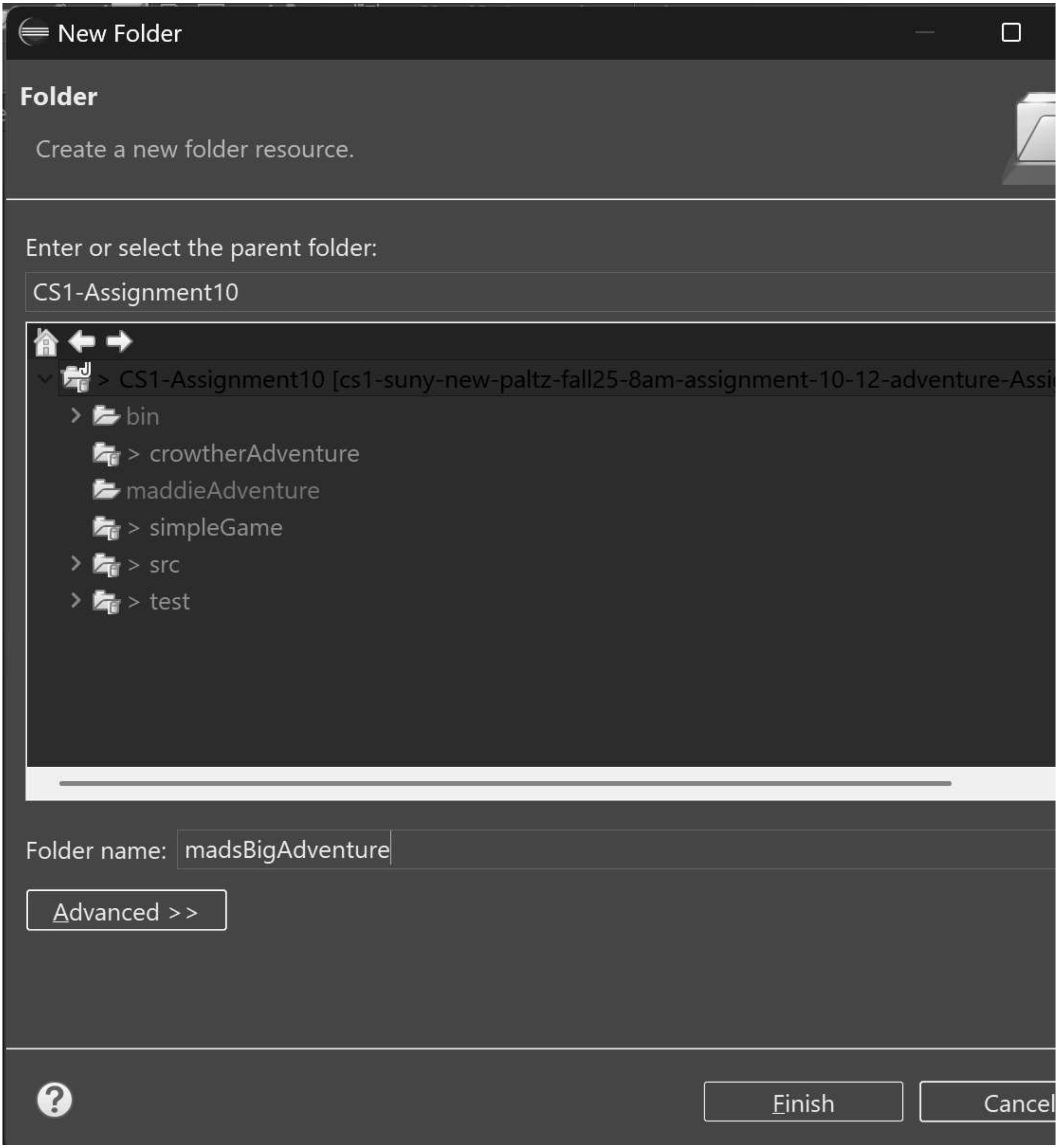
```
> CS1-Assignment10 [cs1-suny-new-paltz-fall25-8am-assignment-10-12-adventure-Assign-
    > src
        > adventure
            > AdventureProgram.java
            > Room.java
            > RoomManager.java
    > test
    JRE System Library [jre]
    JUnit 5
    Referenced Libraries
    > crowtherAdventure
        items.txt
        lost.jpg
        title.jpg
        won.jpg
    > simpleGame
        escape.jpg
        items.txt
        lockedDoor.jpg
        room1.jpg
        room2.jpg
        rooms.txt
    AdventureLib.jar
    assignDeps.jar
    checkstyle_config.xml
```

2. Create a new folder called `uniqueGame` at the same level as `src/`


File
Edit
Source
Refactor
Navigate
Search
Project
Run
Window
Help
Package Explorer
JUnit
> CS1-Assignment10 [cs1-suny-new-paltz-fall25-8am-assignment-10-12-adventure-As
> src
> adv
> test
JRE Syst
JUnit 5
Referenc
> crowtl
> simple
Adventu
assignDe
checksty
New >
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W >
Show in Local Terminal >
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+↓
Build Path >
Source Alt+Shift+S >
Refactor Alt+Shift+T >
Import...
Export...
Source >
Refresh F5
Close Project
Assign Working Sets...
Coverage As >
Run As >
Debug As >
Restore from Local History...
Checkstyle >
Team >
Compare With >
Java Project
Maven Proje
Project...
Package
Class
Interface
Enum
Record
Annotation
Source Fold
Java Workin
Folder
File
Untitled Tex
Task
JUnit Test Ca
Other...



Instructions: Right-click on the project name `CS1-Assignment10-12` → `New` → `Folder` → enter `uniqueGame` →`Finish`


New Folder
Folder
Create a new folder resource.
> CS1-Assignment10 [cs1-suny-new-paltz-fall25-8am-assignment-10-12-adventure-Assi
bin
> crowtherAdventure
maddieAdventure
> simpleGame
> src
> test
Advanced >>
Finish
Cancel

Enter or select the parent folder: CS1-Assignment10

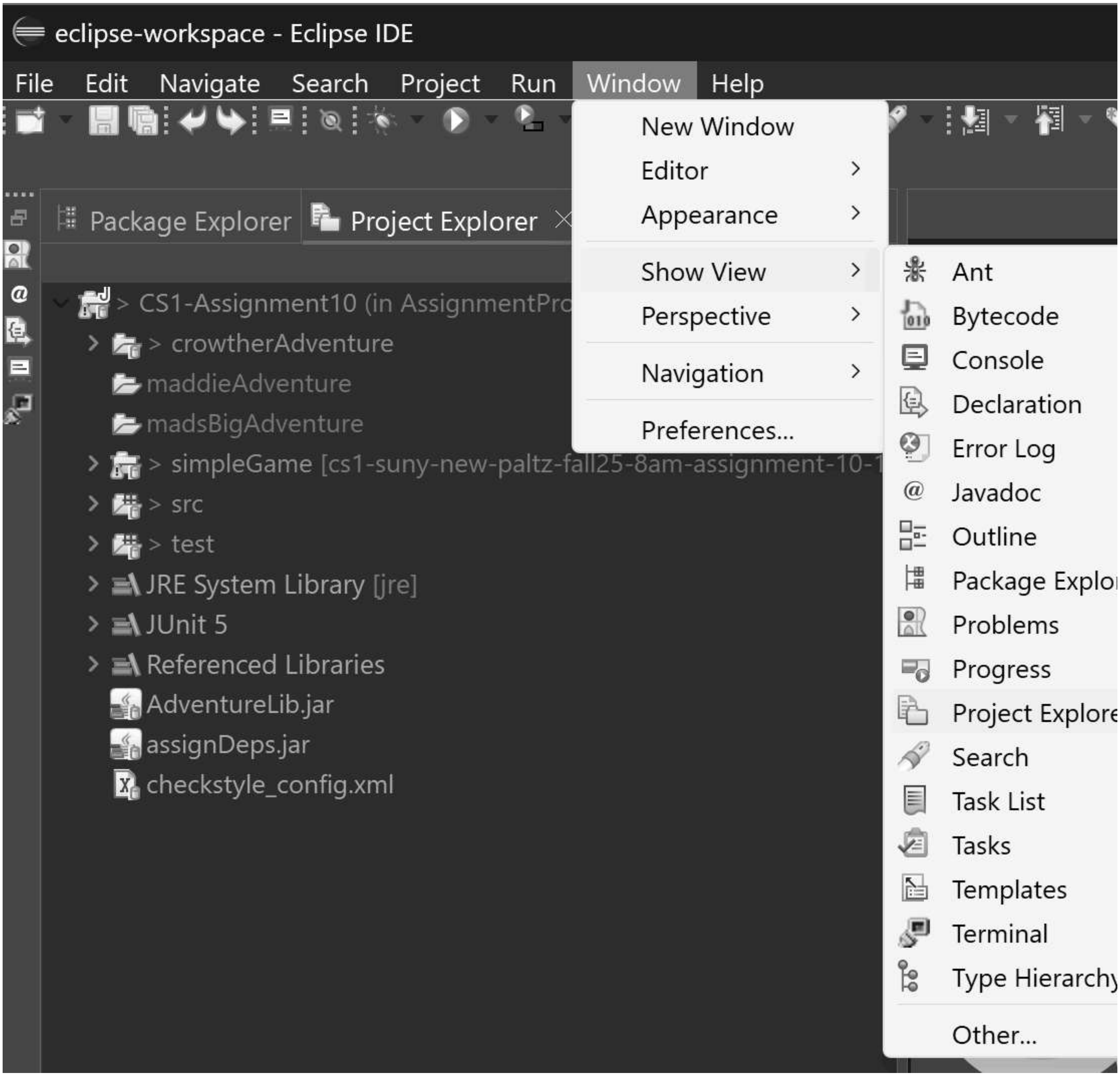
Folder name: madsBigAdventure

2a. If you had difficulty with file creation:

The fix

Switch to **Project Explorer**:

```
Window → Show View → Project Explorer
```



Now Eclipse shows:

- Your game folder, repeat step 2 if not showing up.

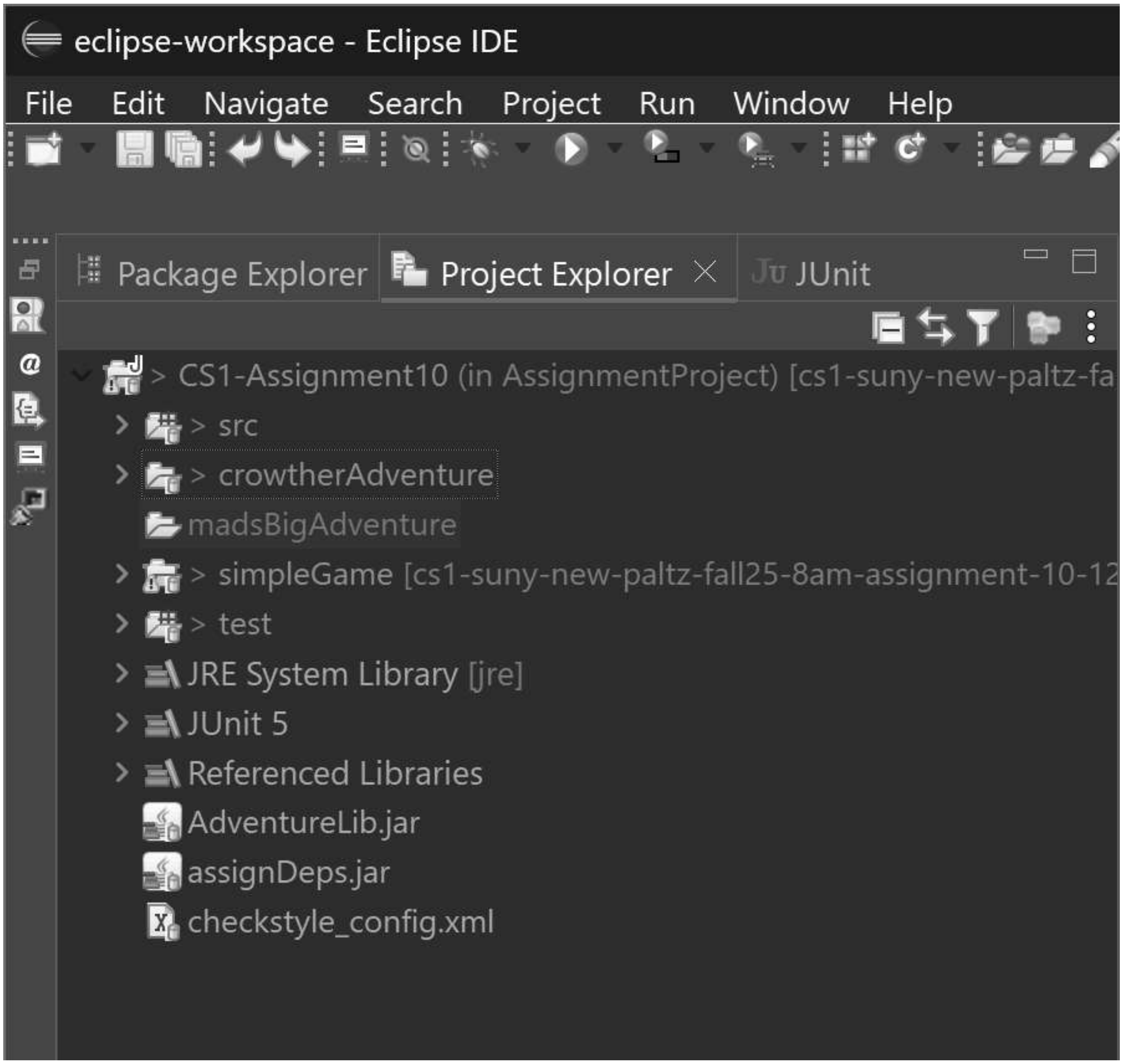
To proceed: Your project's file structure **must** look like this:

```
CS1-Assignment10/
├── src/
│   └── adventure/
│       ├── AdventureProgram.java
│       ├── Room.java
│       └── RoomManager.java (IMPORTANT: there is a new version of this file , see below)
│
├── uniqueGame/    <-- your game folder
│   ├── rooms.txt  <-- we will fill it with unique content, see format below
│   ├── items.txt  <-- we will fill it with unique content, see format below
│   ├── title.jpg  <-- we will copy/paste these .jpg files into this folder for the raw game- when you are
```

```
making your own, you will put them here
│   ├── room2.jpg
│   ├── won.jpg
│   └── lost.jpg
│
├── crowtherAdventure/      (prior example)
├── simpleGame/             (prior example)
├── AdventureLib.jar
└── assignDeps.jar
```

Key rule

- `src/` → **Java code only, with updated RoomManager.java**
- `uniqueGame/` → **where you put your .jpg and .txt files only**
- Your game folder **must NOT be inside `src`, only at the same level.** Here, my sample game folder is called `madsBigAdventure/`, but you can name it whatever you like.



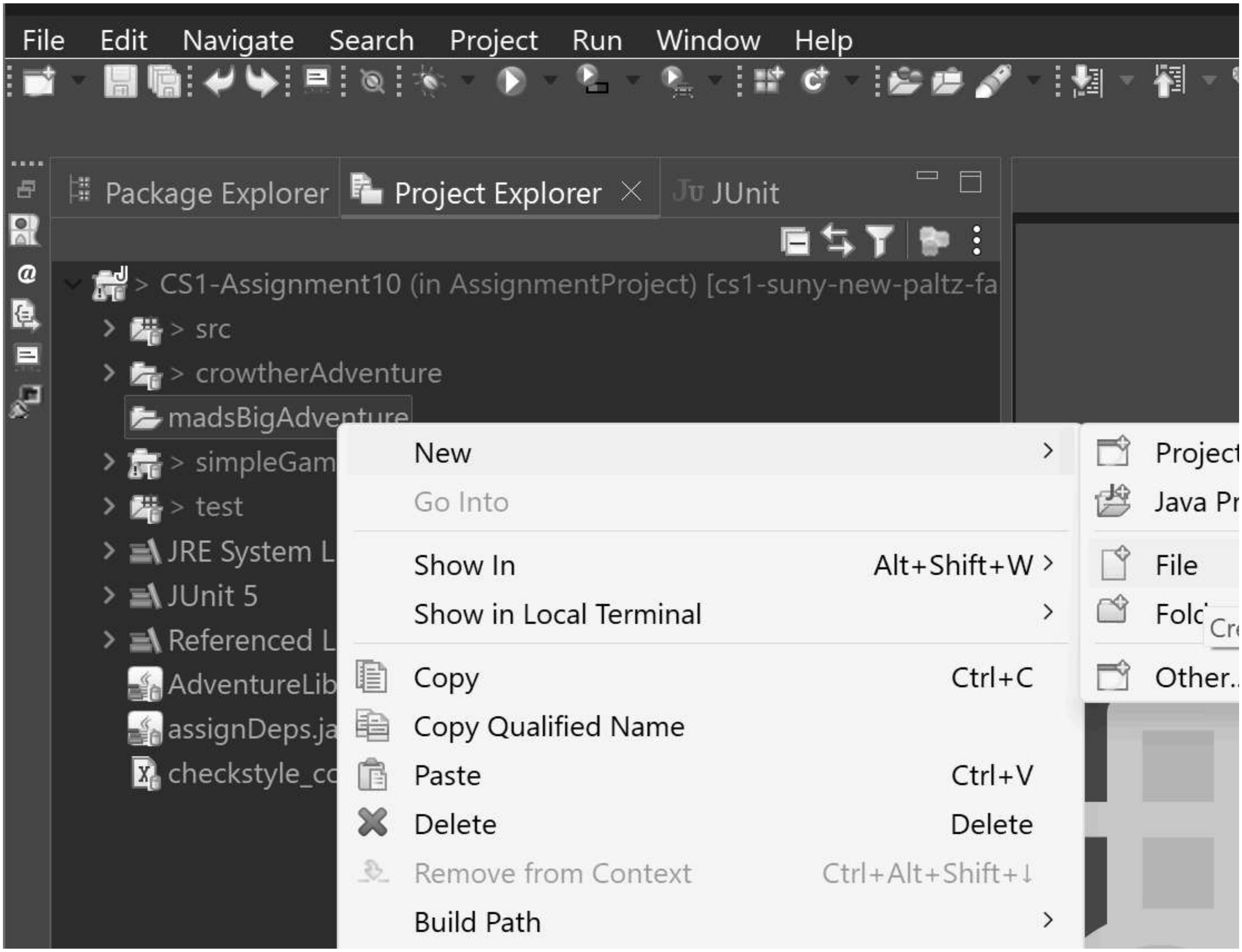
To get the game runnable, first you must copy and paste the .jpg files into your `uniqueGame/` folder. You can use the ones locates already in `crowtherAdventure/`. After the game is runnable, I want you to add your own images, to this exact location (keep the same filenames, or update `rooms.txt` accordingly).

3. Creating Files (Raw Game)

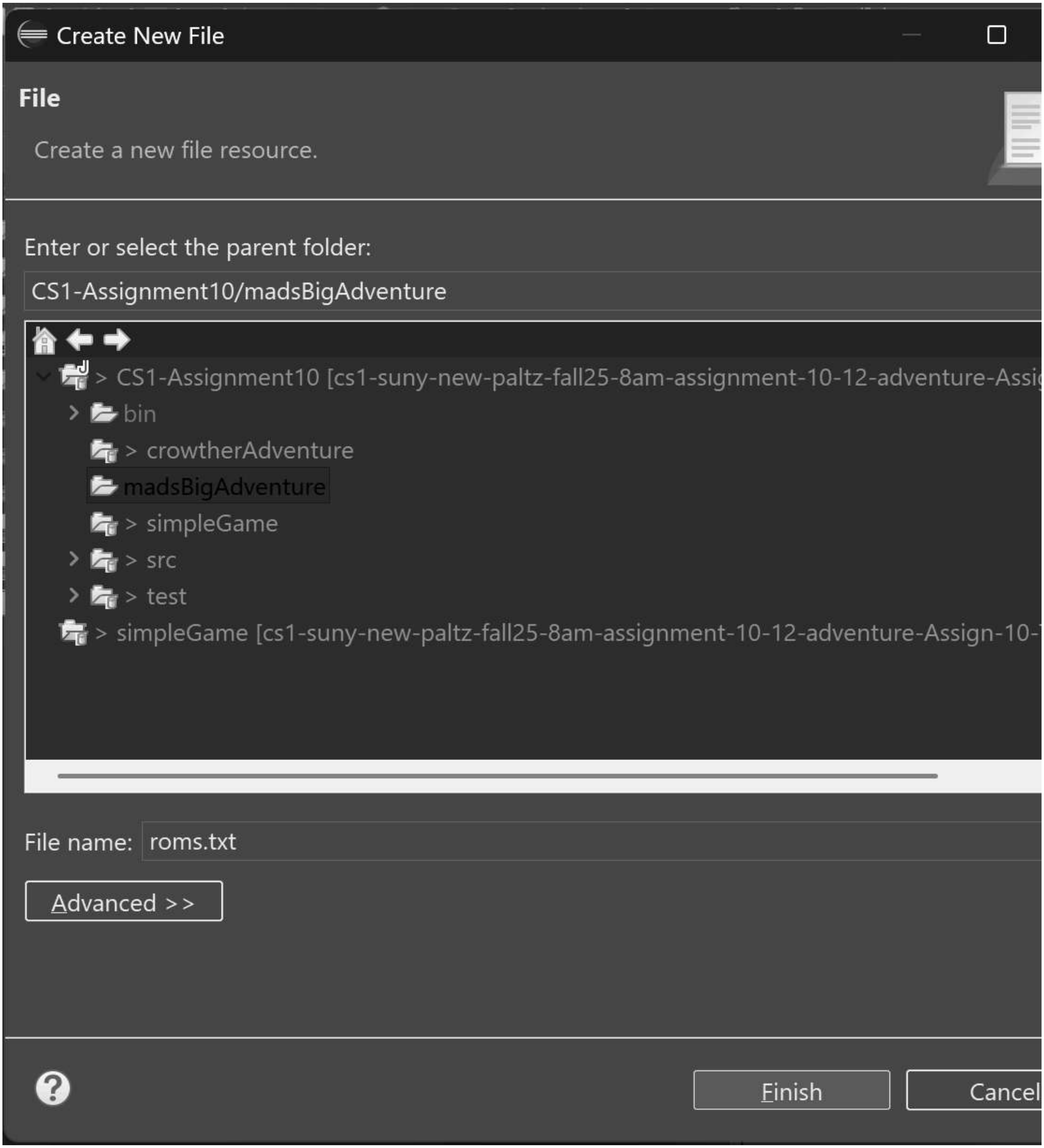
Right-click **uniqueGame** → `New → File`

Inside, create:

- `rooms.txt`
- `items.txt`



Add a new file



Create New File

File

Create a new file resource.

Enter or select the parent folder:

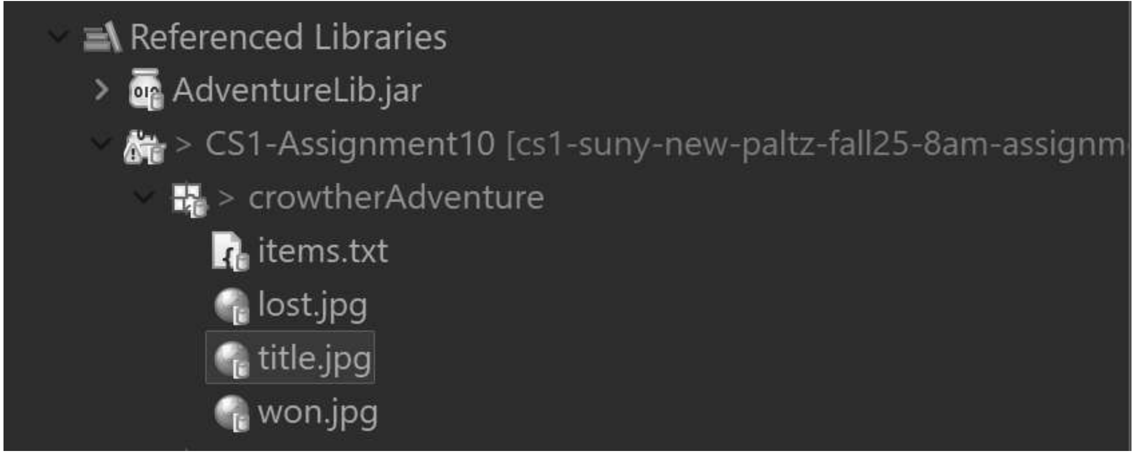
CS1-Assignment10/madsBigAdventure

> CS1-Assignment10 [cs1-suny-new-paltz-fall25-8am-assignment-10-12-adventure-Assig
bin
> crowtherAdventure
madsBigAdventure
> simpleGame
> src
> test
> simpleGame [cs1-suny-new-paltz-fall25-8am-assignment-10-12-adventure-Assign-10-

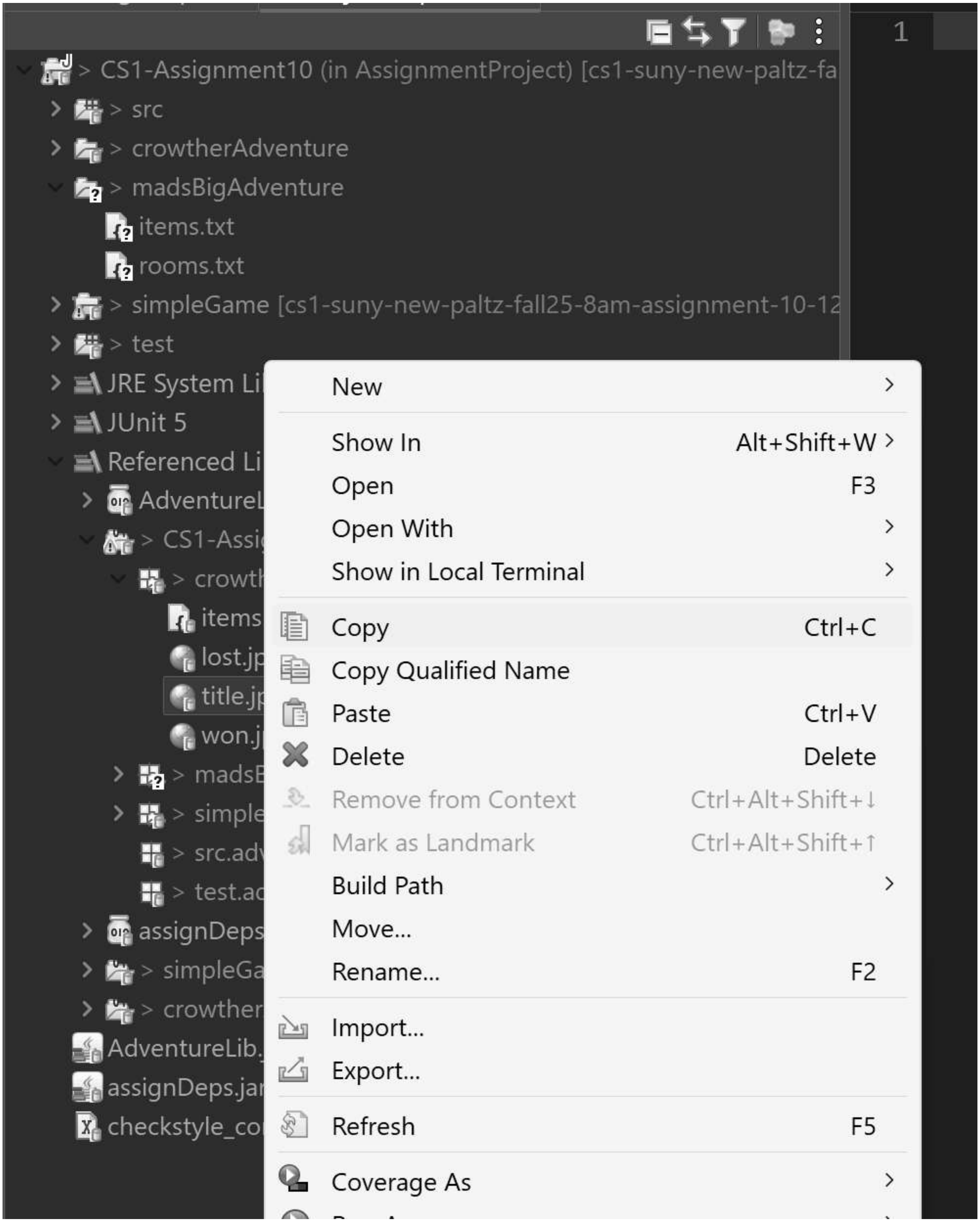
File name: roms.txt

Advanced >>
Finish
Cancel

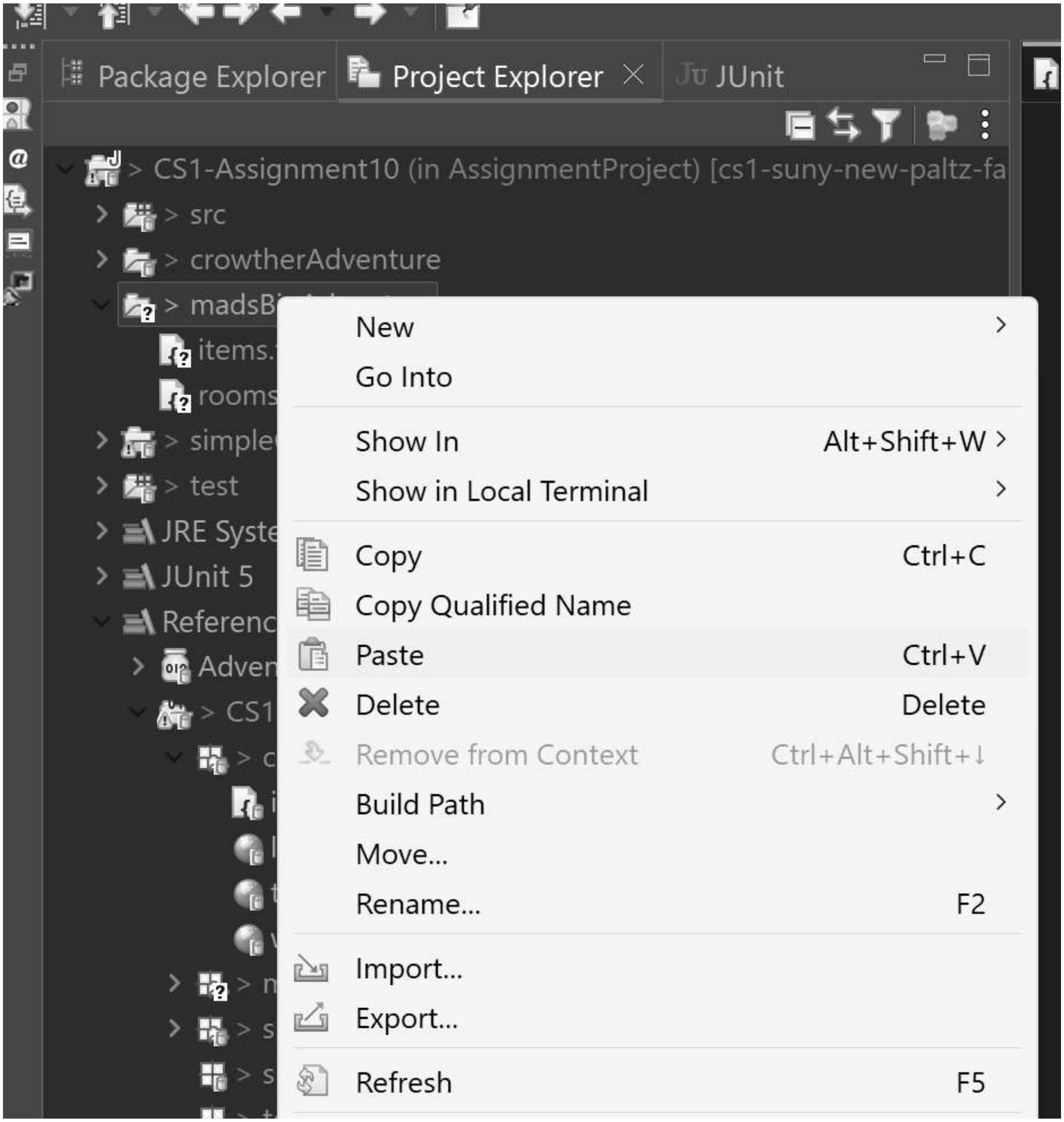
call it rooms.txt. Do the *same exact process* again to create items.txt. Now, once you did that, go inside the crowtherAdventure folder to get the starter images copied (title, win, lost, etc.)



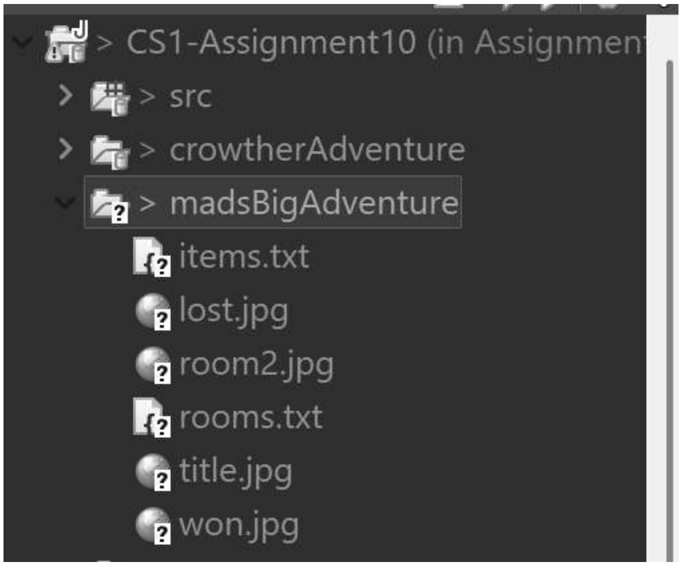
- Referenced Libraries
  - AdventureLib.jar
  - > CS1-Assignment10 [cs1-suny-new-paltz-fall25-8am-assignm
    - > crowtherAdventure
      - items.txt
      - lost.jpg
      - title.jpg
      - won.jpg


1
> CS1-Assignment10 (in AssignmentProject) [cs1-suny-new-paltz-fa
> src
> crowtherAdventure
> madsBigAdventure
items.txt
rooms.txt
> simpleGame [cs1-suny-new-paltz-fall25-8am-assignment-10-12
> test
JRE System Li
JUnit 5
Referenced Li
AdventureL
> CS1-Assi
> crowth
items
lost.jp
title.jp
won.j
> madsE
> simple
> src.adv
> test.ac
assignDeps
> simpleGa
> crowther
AdventureLib.
assignDeps.jar
checkstyle_co
New
Show In Alt+Shift+W
Open F3
Open With
Show in Local Terminal
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+↓
Mark as Landmark Ctrl+Alt+Shift+↑
Build Path
Move...
Rename... F2
Import...
Export...
Refresh F5
Coverage As

copy it



paste inside your game folder



now your structure should look like this, with these exact files to start with.

4. Now, inside uniqueGame, we will do rooms.txt:

Correct `rooms.txt` format.

Use this to get the program to run, then modify it later to create your own adventure. DO NOT mess with the spacing!

```
1
1
title.jpg
You are in a small lobby. A hallway leads NORTH. A locked door is EAST.
KEYCARD
NORTH/2
EAST/3/KEYCARD
DOWN/4

2
room2.jpg
You are in the hallway. The lobby is SOUTH.

SOUTH/1

3
won.jpg
You swipe the keycard, the door clicks open, and you escape. You win!

4
lost.jpg
You fall into a padded room labeled "BONK".

FORCED/1
```

Rules for your creativity:

- **Room ID** on first line. This is a number
- **Image filename** on next line (the .jpg)
- **Description** on next line. NOTE: must be 1 single line
- **Item ID required to enter** (or blank) on next line. This is a word that *must* be already in the items.txt file (e.g. KEYCARD)
- **Exits / actions** on next line One action per line, using this format: `ACTION_ID/NEXT_ROOM_ID[/ITEM_REQUIRED]`

Where: `ACTION_ID` → what the player types (e.g. NORTH, EAST) `NEXT_ROOM_ID` → the room number this action leads to `ITEM_REQUIRED` (optional) → item needed to use this action (e.g. EAST/3/KEYCARD)

Inside uniqueGame:

4. Correct `items.txt`

Use this to get the program to run, then modify it later to create your own adventure. Copy it exactly as shown to start. DO NOT mess with the spacing!

```
KEYCARD
a scratched blue keycard that says "LAB"

SUSHI
a freshly made plate of sushi rolls

FISHFOOD
a tiny packet of fish food
```

Rules for your creativity:

- Item ID on one line
- Description on the **next** line
- 1 Blank line between items

---

5. Update the old RoomManager.java

You must replace your old RoomManager.java file inside src/ with this one:

```
package adventure;

import common.CS1Reader;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

/**
 * Manages the collection of rooms in the game and handles data loading.
 */
public class RoomManager {
    private HashMap<String, Room> rooms;
    private HashMap<String, GameItem> items;
    private String startingRoomId;

    /**
     * Creates a new RoomManager based on the data in gameFolder.
     *
     * @param gameFolder the folder containing game data files
     */
    public RoomManager(String gameFolder) {
        rooms = new HashMap<String, Room>();
        items = new HashMap<String, GameItem>();

        String itemsFilePath = gameFolder + "/items.txt";
        String roomsFilePath = gameFolder + "/rooms.txt";

        // Load items first since rooms depend on items.
        loadItems(itemsFilePath);

        // Load rooms
        loadRooms(roomsFilePath);
    }

    /**
     * Returns the Room the game starts in.
     *
     * @return the starting room
     */
    public Room getStartingRoom() {
        return rooms.get(startingRoomId);
    }

    /**
     * Returns the Room corresponding to the given id, or null if no such room exists.
     *
     * @param id the room id to search for
     * @return the Room or null
     */
    public Room getRoom(String id) {
        return rooms.get(id);
    }

    /**
     * Loads items from the items file.
     *
     * @param filename the path to the items file
     */
    private void loadItems(String filename) {
        CS1Reader reader = new CS1Reader(filename);

        String itemLine = reader.readLine();
        while (itemLine != null) {
            if (itemLine.length() > 0) {
                String itemId = itemLine.trim();
                String description = reader.readLine();

                GameItem item = new GameItem(itemId, description);
                items.put(itemId, item);
            }
            itemLine = reader.readLine();
        }

        // DEBUG: confirm items loaded
        System.out.println("Loaded items: " + items.keySet());
    }

    /**
     * Loads rooms from the rooms file.
     *
     * @param filename the path to the rooms file
     */
    private void loadRooms(String filename) {
        CS1Reader reader = new CS1Reader(filename);

        // First line is the starting room ID
        startingRoomId = reader.readLine();

        // Read each room (line holds the roomId)
        String line = reader.readLine();
        while (line != null) {
            if (line.length() > 0) {
                // Room ID
                String roomId = line.trim();

                // Image file
                String imageFile = reader.readLine();

                // Description
                String description = reader.readLine();

                // Items in room (comma-separated or empty line)
                String itemLine = reader.readLine();
                if (itemLine == null) {
                    itemLine = "";
                }
                itemLine = itemLine.trim();

                // DEBUG (optional): show what item line was read
                System.out.println("Room " + roomId + " itemLine=[" + itemLine + "]");

                List<GameItem> roomItems = new ArrayList<GameItem>();
                if (itemLine.length() > 0) {
                    String[] itemIds = itemLine.split(",");
                    for (int i = 0; i < itemIds.length; i++) {
                        String itemId = itemIds[i].trim();
                        GameItem item = items.get(itemId);
                        if (item != null) {
                            roomItems.add(item);
                        } else {
                            // DEBUG (optional): if this prints, the ID didn't match items.txt
                            System.out.println("WARNING: Room " + roomId + " references unknown item [" +
itemId + "]");
                        }
                    }
                }

                // Actions
                List<GameAction> actions = new ArrayList<GameAction>();
                line = reader.readLine();

                while (line != null && line.trim().length() > 0) {
                    line = line.trim();

                    // Safety: action lines must contain at least "ID/NEXTROOM"
                    String[] parts = line.split("/");
                    if (parts.length < 2) {
                        System.out.println("BAD ACTION LINE (missing '/'): [" + line + "]");
                        line = reader.readLine();
                        continue;
                    }

                    String actionId = parts[0];
                    String nextRoom = parts[1];

                    ArrayList<String> requirements = new ArrayList<String>();
                    for (int i = 2; i < parts.length; i++) {
                        requirements.add(parts[i].trim());
                    }

                    actions.add(new GameAction(actionId, nextRoom, requirements));
                    line = reader.readLine();
                }

                // Create and store room
                Room room = new Room(imageFile, description, roomItems, actions);
                rooms.put(roomId, room);

                // CRITICAL: move to the next roomId after the blank separator line
                line = reader.readLine();

            } else {
                // Empty line, read next
                line = reader.readLine();
            }
        }

        // DEBUG: confirm rooms loaded
        System.out.println("Loaded rooms: " + rooms.keySet());
        System.out.println("Starting room: " + startingRoomId);
    }
}
```

6. Raw Game complete: How to Run it

1. Right-click `AdventureProgram.java`
2. `Run As → Java Application`
3. When prompted:

   ```
   Enter the folder to load for a game of Adventure:
   uniqueGame
   ```

---

7. How to Play the Raw Game (Commands) (My example, madsBigAdventure -- *please* adapt to your own)

Valid commands:

```
NORTH
SOUTH
EAST
```

```
DOWN
GET KEYCARD
LIST
QUIT
```

Winning path

```
GET KEYCARD
EAST
```

If an item says “not here”

- You’re in the wrong room
- Use movement commands first

---

8. Play with it a bunch to get the hang of it.

I recommend you test the raw game first, then you modify `rooms.txt` and `items.txt` to create your own unique adventure. Find images online to use as room images, and place them in the `uniqueGame/` folder. Make sure to update the filenames in `rooms.txt` accordingly! (e.g room2 might become forest.jpg if you want a forest scene).

```
Note:  Read through the assignment pdf, and remember that you'll need:
    - at least 3 rooms,
    - 1 item,
    - 1 transition that requires an item, and
    - 1 transition that doesn’t require an item
```