Java Capstone Project

Compact Programming Course - Java

Group#10

1. Bhavana Shivaraju [7221863]
2. Mohammad Ali Moradi [7221791]
3. Aftab Makbul Makandar [7222045]

Agenda

Problem Statement and Tasks

Requirements to the system

Class Diagram

Component Diagram

Project Overview

Management I/O in the system

User interfaces

Concurrency

Results of the Unit Test of the System

Smart House: Simulation of the Energy supply and consumption

Tasks

- Simulation of the Smart Objects
- Simulation of the Energy Sources
- Management system for the house consumption
- Design of the User Interface for the Managing Smart Objects
- Design of the User Interface for Managing Energy Sources

Agenda

Problem Statement and Tasks

Requirements to the system

Class Diagram

Component Diagram

Project Overview

Management I/O in the system

User interfaces

Concurrency

Results of the Unit Test of the System

System Requirements

Hardware Requirements

•**Processor**: Dual-core processor (e.g., Intel Core i3 or AMD equivalent).

•**RAM**: 4 GB.

•**Storage**: 2 GB of free disk space (for Java Development Kit (JDK), Integrated Development Environment (IDE), and project files).

•**Graphics**: Basic integrated graphics (no special GPU needed for Java programs).

Software Requirements

Operating System: Windows, macOS

Java Development Kit (JDK)

Integrated Development Environment (IDE): IntelliJ IDEA, Eclipse, or NetBeans

Agenda

Problem Statement and Tasks

Requirements to the system

Class Diagram

Component Diagram

Project Overview

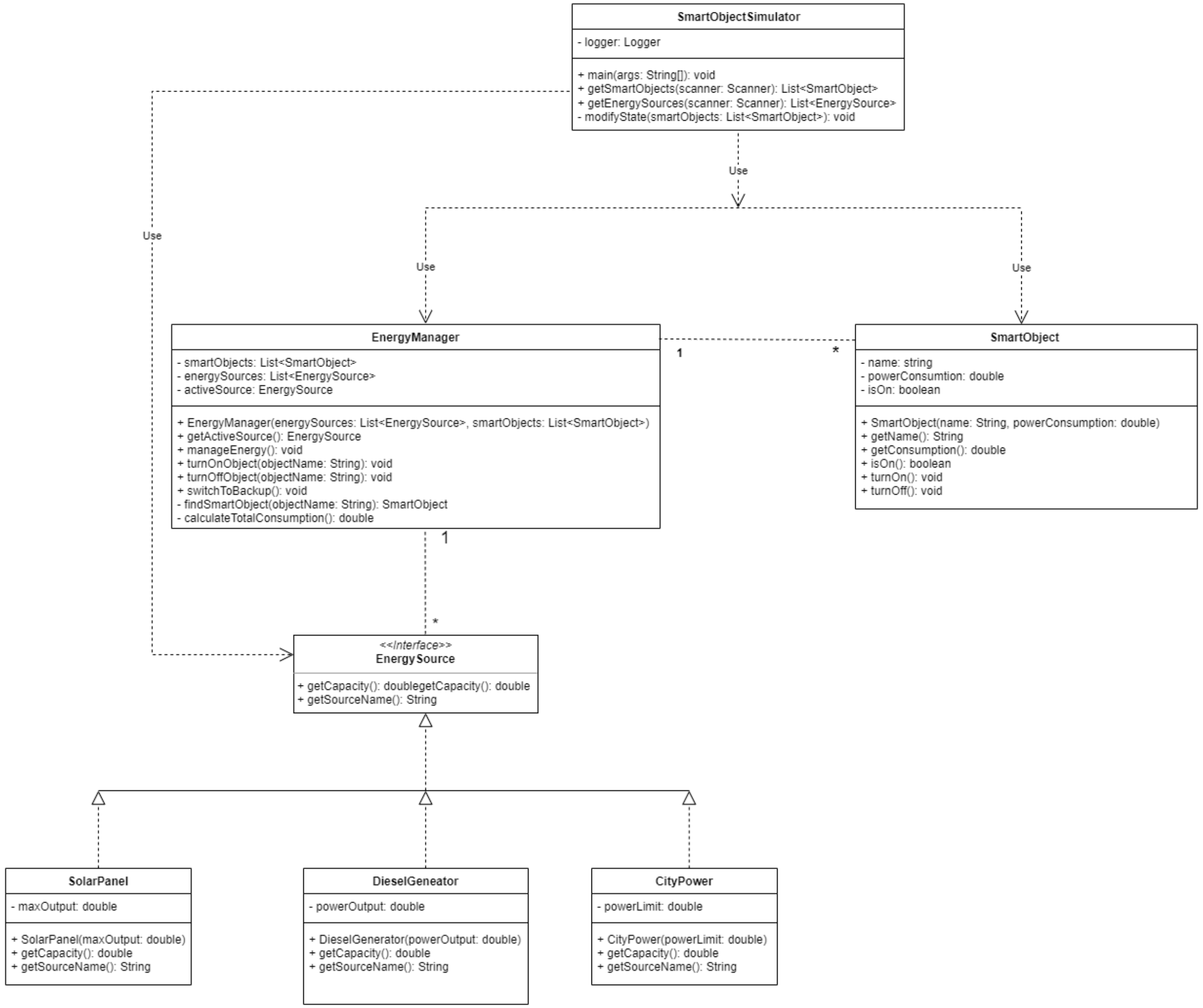
Management I/O in the system

User interfaces

Concurrency

Results of the Unit Test of the System

SmartHouseSimulation



SmartObjectSimulator

- logger: Logger

+ main(args: String[]): void
+ getSmartObjects(scanner: Scanner): List<SmartObject>
+ getEnergySources(scanner: Scanner): List<EnergySource>
- modifyState(smartObjects: List<SmartObject>): void

EnergyManager

- smartObjects: List<SmartObject>
- energySources: List<EnergySource>
- activeSource: EnergySource

+ EnergyManager(energySources: List<EnergySource>, smartObjects: List<SmartObject>)
+ getActiveSource(): EnergySource
+ manageEnergy(): void
+ turnOnObject(objectName: String): void
+ turnOffObject(objectName: String): void
+ switchToBackup(): void
- findSmartObject(objectName: String): SmartObject
- calculateTotalConsumption(): double

SmartObject

- name: string
- powerConsumtion: double
- isOn: boolean

+ SmartObject(name: String, powerConsumption: double)
+ getName(): String
+ getConsumption(): double
+ isOn(): boolean
+ turnOn(): void
+ turnOff(): void

<<Interface>>
EnergySource

+ getCapacity(): doublegetCapacity(): double
+ getSourceName(): String

SolarPanel

- maxOutput: double

+ SolarPanel(maxOutput: double)
+ getCapacity(): double
+ getSourceName(): String

DieselGeneator

- powerOutput: double

+ DieselGenerator(powerOutput: double)
+ getCapacity(): double
+ getSourceName(): String

CityPower

- powerLimit: double

+ CityPower(powerLimit: double)
+ getCapacity(): double
+ getSourceName(): String

Agenda

Problem Statement and Tasks

Requirements to the system

Class Diagram

Component Diagram

Project Overview

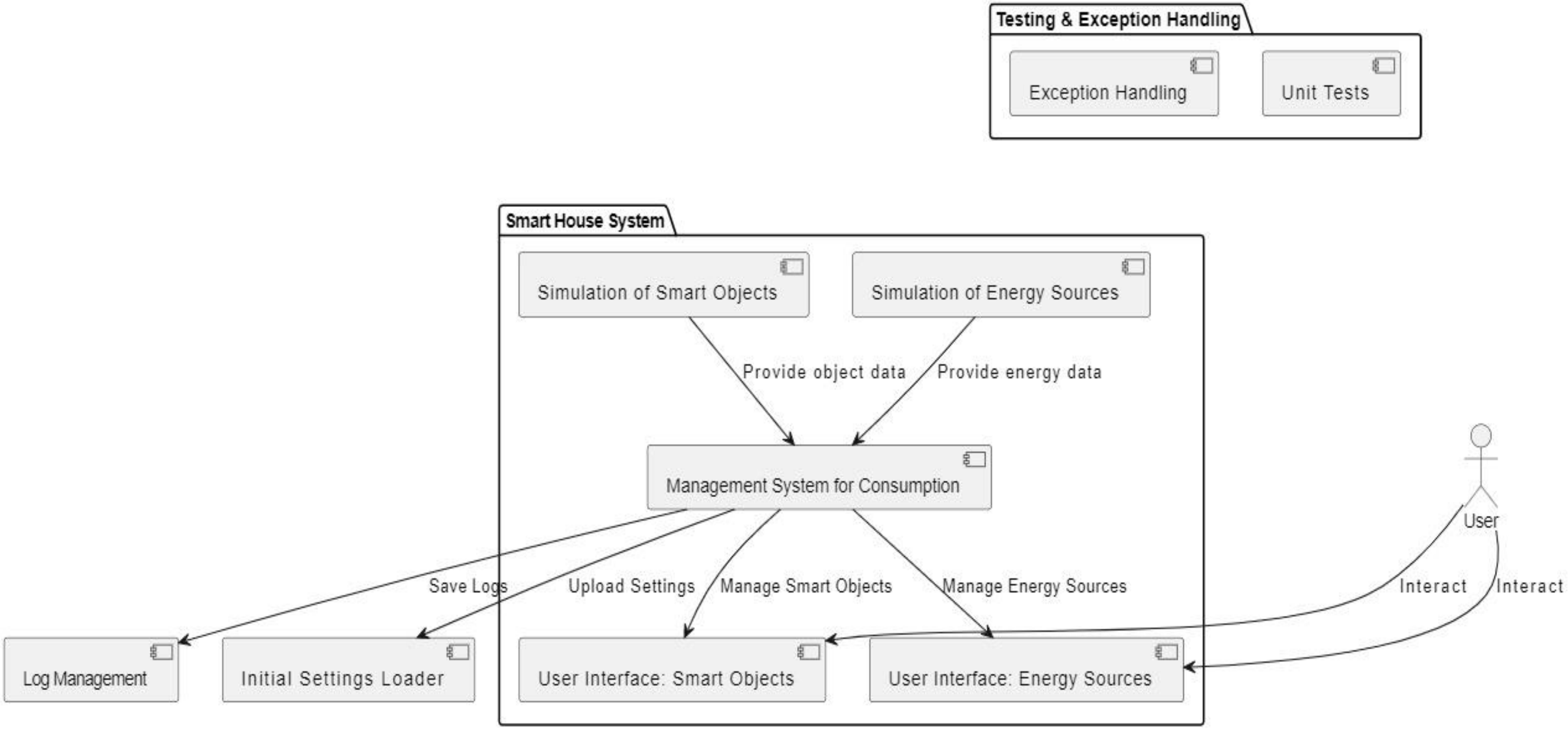
Management I/O in the system

User interfaces

Concurrency

Results of the Unit Test of the System

Component Diagram for Smart House Simulation



Agenda

Problem Statement and Tasks

Requirements to the system

Class Diagram

Component Diagram

Project Overview

Management I/O in the system

User interfaces

Concurrency

Results of the Unit Test of the System

Management I/O in the system

The System handles I/p and O/p for various operations across different methods, enabling user interactions like adding or removing Smart Objects and managing Energy Sources.

1.Energy Manager

➔ User mange energy resources.

➔ List Energy sources is displayed.

2. Smart Objects

➔ Users can add new smart objects and as well remove.

➔ List of smart objects is displayed

Agenda

Problem Statement and Tasks

Requirements to the system

Class Diagram

Component Diagram

Project Overview

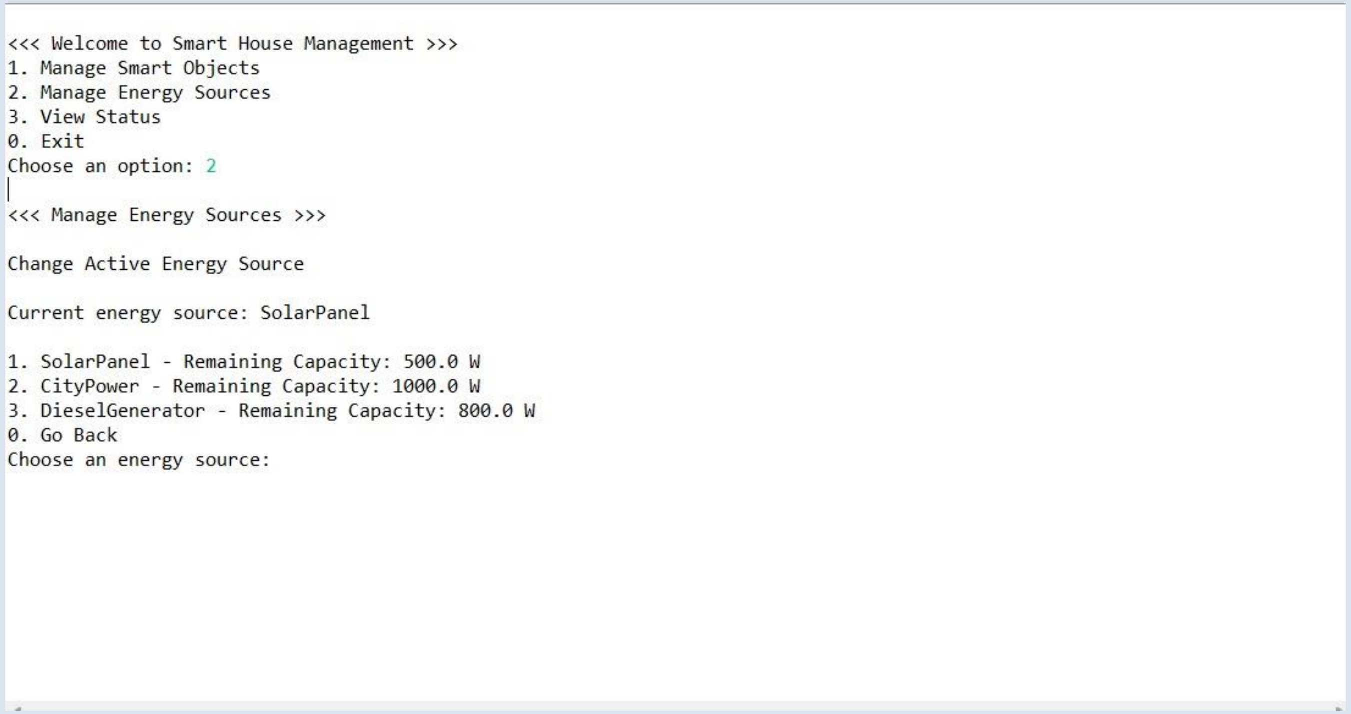
Management I/O in the system

User interfaces

Concurrency

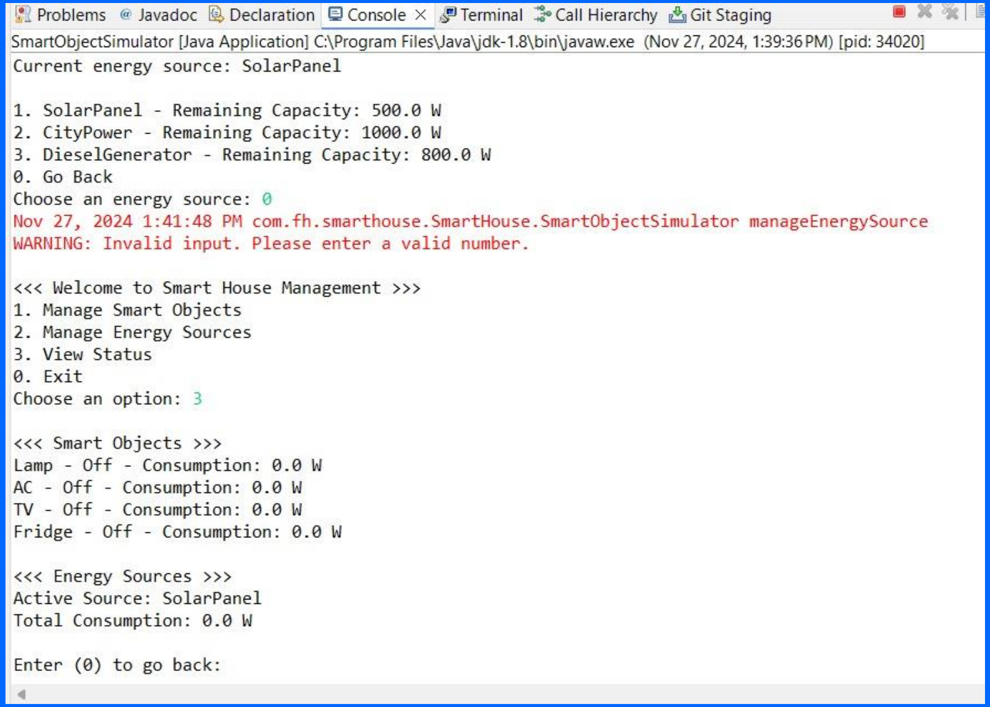
Results of the Unit Test of the System

Available Energy Sources

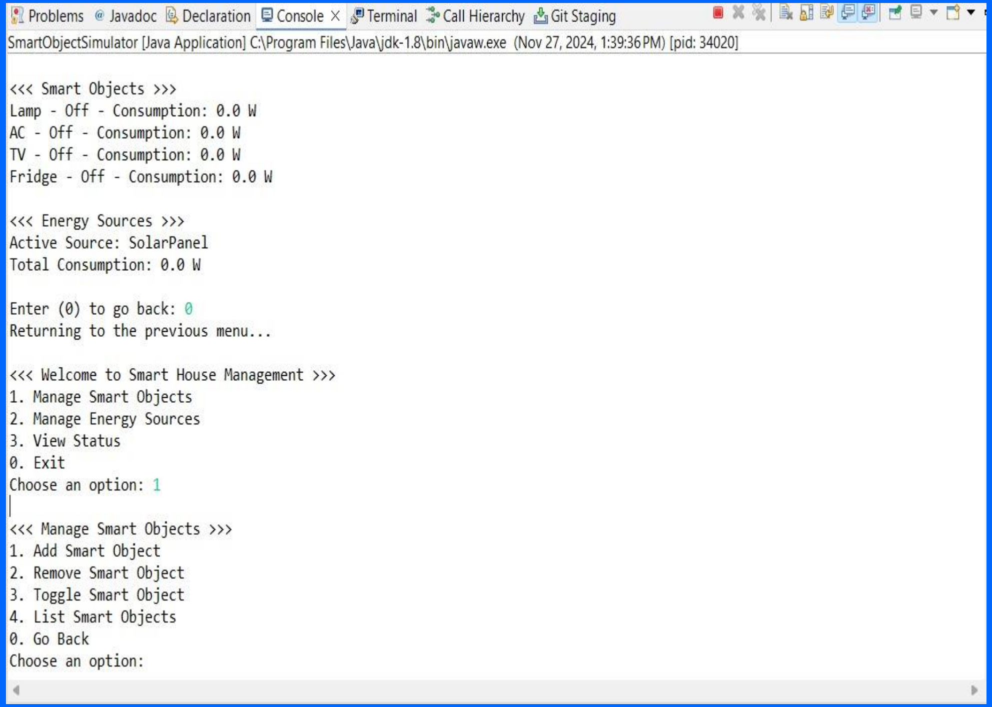


```
<<< Welcome to Smart House Management >>>
1. Manage Smart Objects
2. Manage Energy Sources
3. View Status
0. Exit
Choose an option: 2

<<< Manage Energy Sources >>>

Change Active Energy Source

Current energy source: SolarPanel

1. SolarPanel - Remaining Capacity: 500.0 W
2. CityPower - Remaining Capacity: 1000.0 W
3. DieselGenerator - Remaining Capacity: 800.0 W
0. Go Back
Choose an energy source:
```

UI for EnergySources and SmartObjects



```
SmartObjectSimulator [Java Application] C:\Program Files\Java\jdk-1.8\bin\javaw.exe  (Nov 27, 2024, 1:39:36 PM) [pid: 34020]
Current energy source: SolarPanel

1. SolarPanel - Remaining Capacity: 500.0 W
2. CityPower - Remaining Capacity: 1000.0 W
3. DieselGenerator - Remaining Capacity: 800.0 W
0. Go Back
Choose an energy source: 0
Nov 27, 2024 1:41:48 PM com.fh.smarthouse.SmartHouse.SmartObjectSimulator manageEnergySource
WARNING: Invalid input. Please enter a valid number.

<<< Welcome to Smart House Management >>>
1. Manage Smart Objects
2. Manage Energy Sources
3. View Status
0. Exit
Choose an option: 3

<<< Smart Objects >>>
Lamp - Off - Consumption: 0.0 W
AC - Off - Consumption: 0.0 W
TV - Off - Consumption: 0.0 W
Fridge - Off - Consumption: 0.0 W

<<< Energy Sources >>>
Active Source: SolarPanel
Total Consumption: 0.0 W

Enter (0) to go back:
```



```
SmartObjectSimulator [Java Application] C:\Program Files\Java\jdk-1.8\bin\javaw.exe  (Nov 27, 2024, 1:39:36 PM) [pid: 34020]

<<< Smart Objects >>>
Lamp - Off - Consumption: 0.0 W
AC - Off - Consumption: 0.0 W
TV - Off - Consumption: 0.0 W
Fridge - Off - Consumption: 0.0 W

<<< Energy Sources >>>
Active Source: SolarPanel
Total Consumption: 0.0 W

Enter (0) to go back: 0
Returning to the previous menu...

<<< Welcome to Smart House Management >>>
1. Manage Smart Objects
2. Manage Energy Sources
3. View Status
0. Exit
Choose an option: 1

<<< Manage Smart Objects >>>
1. Add Smart Object
2. Remove Smart Object
3. Toggle Smart Object
4. List Smart Objects
0. Go Back
Choose an option:
```

Agenda

Problem Statement and Tasks

Requirements to the system

Class Diagram

Component Diagram

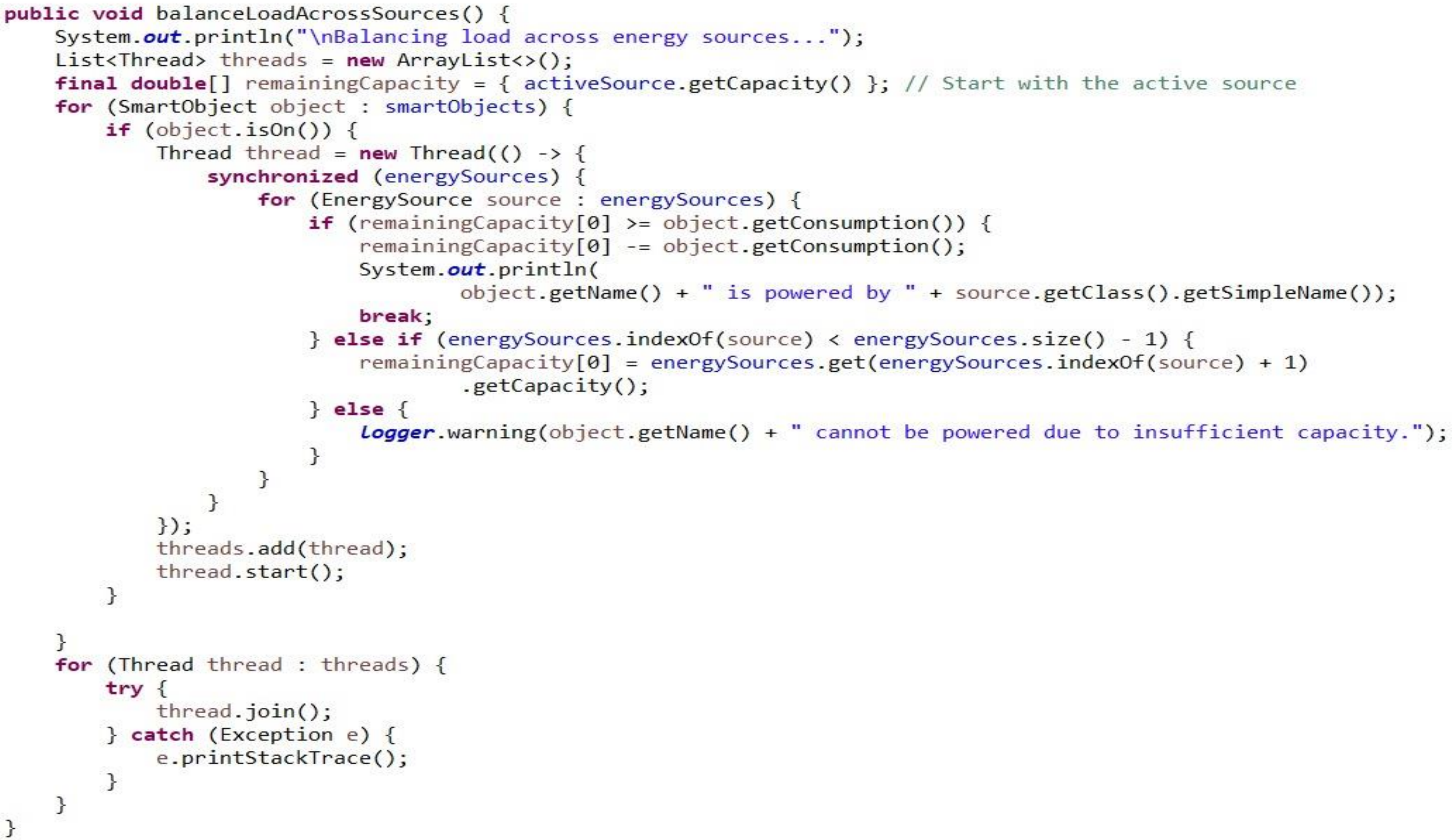
Project Overview

Management I/O in the system

User interfaces

Concurrency

Results of the Unit Test of the System



```
public void balanceLoadAcrossSources() {
    System.out.println("\nBalancing load across energy sources...");
    List<Thread> threads = new ArrayList<>();
    final double[] remainingCapacity = { activeSource.getCapacity() }; // Start with the active source
    for (SmartObject object : smartObjects) {
        if (object.isOn()) {
            Thread thread = new Thread(() -> {
                synchronized (energySources) {
                    for (EnergySource source : energySources) {
                        if (remainingCapacity[0] >= object.getConsumption()) {
                            remainingCapacity[0] -= object.getConsumption();
                            System.out.println(
                                    object.getName() + " is powered by " + source.getClass().getSimpleName());
                            break;
                        } else if (energySources.indexOf(source) < energySources.size() - 1) {
                            remainingCapacity[0] = energySources.get(energySources.indexOf(source) + 1)
                                    .getCapacity();
                        } else {
                            Logger.warning(object.getName() + " cannot be powered due to insufficient capacity.");
                        }
                    }
                }
            });
            threads.add(thread);
            thread.start();
        }

    }
    for (Thread thread : threads) {
        try {
            thread.join();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Agenda

Problem Statement and Tasks

Requirements to the system

Class Diagram

Component Diagram

Project Overview

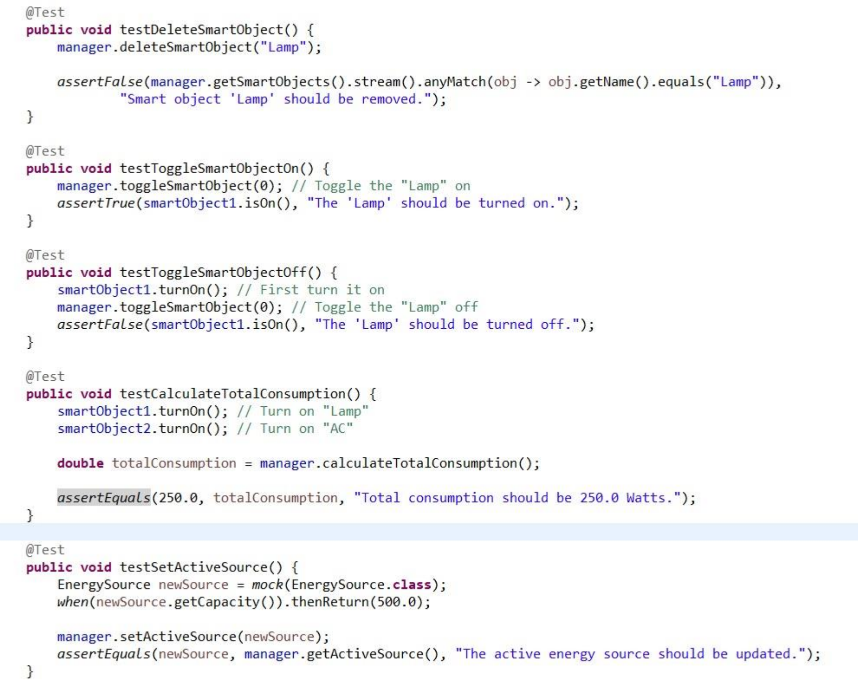
Management I/O in the system

User interfaces

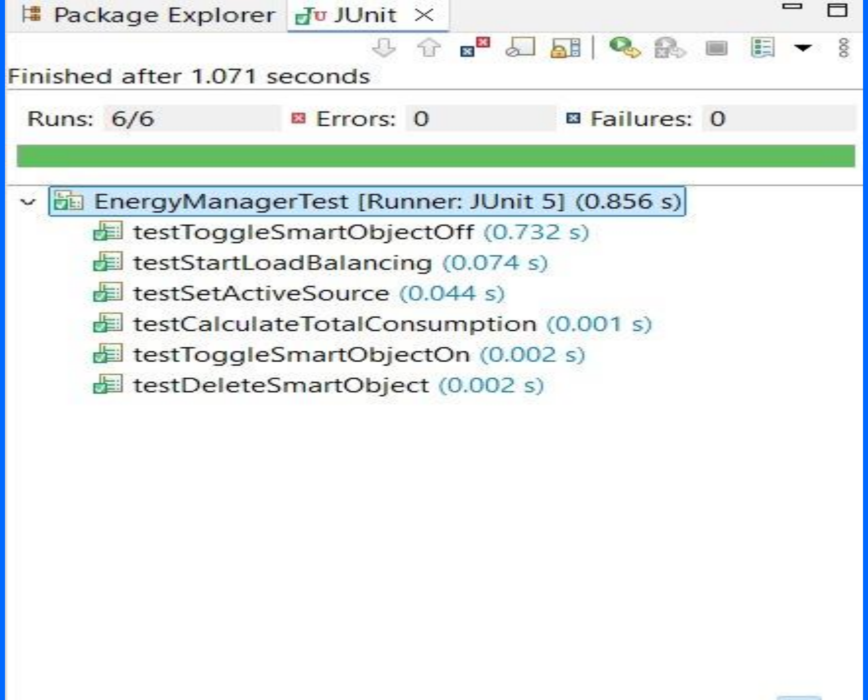
Concurrency

Results of the Unit Test of the System

Junit Test Case for EnergyManager

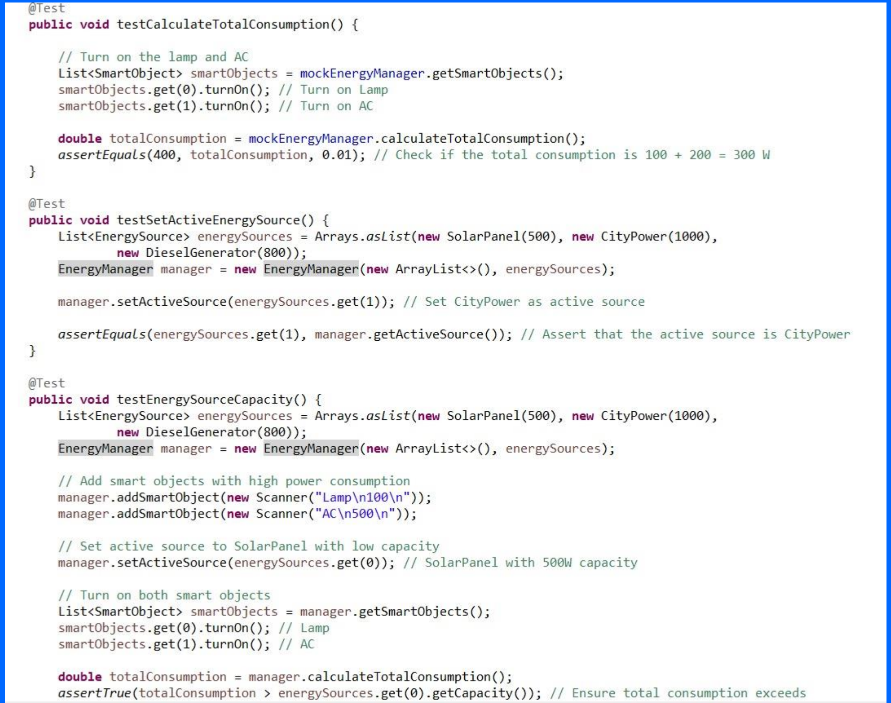


```java
@Test
public void testDeleteSmartObject() {
    manager.deleteSmartObject("Lamp");

    assertFalse(manager.getSmartObjects().stream().anyMatch(obj -> obj.getName().equals("Lamp")),
            "Smart object 'Lamp' should be removed.");
}

@Test
public void testToggleSmartObjectOn() {
    manager.toggleSmartObject(0); // Toggle the "Lamp" on
    assertTrue(smartObject1.isOn(), "The 'Lamp' should be turned on.");
}

@Test
public void testToggleSmartObjectOff() {
    smartObject1.turnOn(); // First turn it on
    manager.toggleSmartObject(0); // Toggle the "Lamp" off
    assertFalse(smartObject1.isOn(), "The 'Lamp' should be turned off.");
}

@Test
public void testCalculateTotalConsumption() {
    smartObject1.turnOn(); // Turn on "Lamp"
    smartObject2.turnOn(); // Turn on "AC"

    double totalConsumption = manager.calculateTotalConsumption();

    assertEquals(250.0, totalConsumption, "Total consumption should be 250.0 Watts.");
}

@Test
public void testSetActiveSource() {
    EnergySource newSource = mock(EnergySource.class);
    when(newSource.getCapacity()).thenReturn(500.0);

    manager.setActiveSource(newSource);
    assertEquals(newSource, manager.getActiveSource(), "The active energy source should be updated.");
}
```

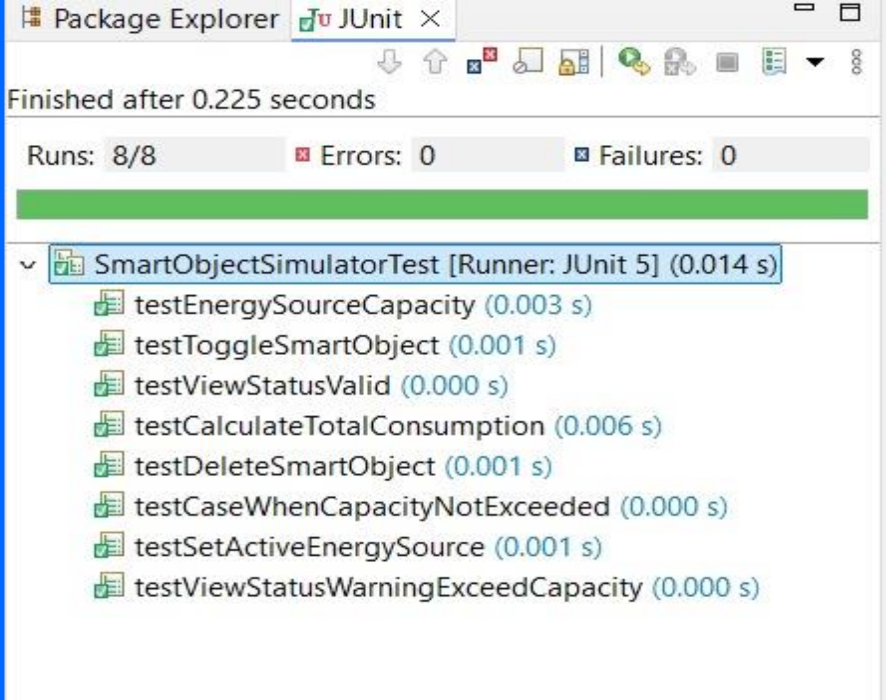


```
Finished after 1.071 seconds

Runs: 6/6    Errors: 0    Failures: 0

EnergyManagerTest [Runner: JUnit 5] (0.856 s)
    testToggleSmartObjectOff (0.732 s)
    testStartLoadBalancing (0.074 s)
    testSetActiveSource (0.044 s)
    testCalculateTotalConsumption (0.001 s)
    testToggleSmartObjectOn (0.002 s)
    testDeleteSmartObject (0.002 s)
```

Junit Test Case for SmartObjectSimulator



```
@Test
public void testCalculateTotalConsumption() {

    // Turn on the lamp and AC
    List<SmartObject> smartObjects = mockEnergyManager.getSmartObjects();
    smartObjects.get(0).turnOn(); // Turn on Lamp
    smartObjects.get(1).turnOn(); // Turn on AC

    double totalConsumption = mockEnergyManager.calculateTotalConsumption();
    assertEquals(400, totalConsumption, 0.01); // Check if the total consumption is 100 + 200 = 300 W
}

@Test
public void testSetActiveEnergySource() {
    List<EnergySource> energySources = Arrays.asList(new SolarPanel(500), new CityPower(1000),
            new DieselGenerator(800));
    EnergyManager manager = new EnergyManager(new ArrayList<>(), energySources);

    manager.setActiveSource(energySources.get(1)); // Set CityPower as active source

    assertEquals(energySources.get(1), manager.getActiveSource()); // Assert that the active source is CityPower
}

@Test
public void testEnergySourceCapacity() {
    List<EnergySource> energySources = Arrays.asList(new SolarPanel(500), new CityPower(1000),
            new DieselGenerator(800));
    EnergyManager manager = new EnergyManager(new ArrayList<>(), energySources);

    // Add smart objects with high power consumption
    manager.addSmartObject(new Scanner("Lamp\n100\n"));
    manager.addSmartObject(new Scanner("AC\n500\n"));

    // Set active source to SolarPanel with low capacity
    manager.setActiveSource(energySources.get(0)); // SolarPanel with 500W capacity

    // Turn on both smart objects
    List<SmartObject> smartObjects = manager.getSmartObjects();
    smartObjects.get(0).turnOn(); // Lamp
    smartObjects.get(1).turnOn(); // AC

    double totalConsumption = manager.calculateTotalConsumption();
    assertTrue(totalConsumption > energySources.get(0).getCapacity()); // Ensure total consumption exceeds
```



Finished after 0.225 seconds

Runs: 8/8 Errors: 0 Failures: 0

```
SmartObjectSimulatorTest [Runner: JUnit 5] (0.014 s)
    testEnergySourceCapacity (0.003 s)
    testToggleSmartObject (0.001 s)
    testViewStatusValid (0.000 s)
    testCalculateTotalConsumption (0.006 s)
    testDeleteSmartObject (0.001 s)
    testCaseWhenCapacityNotExceeded (0.000 s)
    testSetActiveEnergySource (0.001 s)
    testViewStatusWarningExceedCapacity (0.000 s)
```

Thank you