**CSC272 Advance Java Programming – Module 1 Assignment**

View Assignment and code on Github:

<https://github.com/bankszach/csc272-module1-homework.git>

|  |  |
| --- | --- |
| **First Name** | Zachary |
| **Last Name** | Banks |
| **ID#** | 900648740 |
| **Assignment Week#** | 1 |
| **Email Address** | z.banks8740@student.nu.edu |

# How to submit your Assignment

After filling all the parts in this file, please follow the following steps.

1. Add your name and ID to the first page.
2. Save the file in the original format (Docx or Doc)

(Please **do not** convert to other file formats e.g. PDF, ZIP, RAR, …).

1. Rename the file as

*YOUR* ***First*** *Name - YOUR* ***Last*** *Name- YOUR student ID-* CSC272*.docx*

**Example:**

John – Smith - 234566435 – CSC272.docx

1. Upload the file and submit it (only using Blackboard)

Please do not hesitate to contact your instructor for CSC272, should you have any questions.

# Problem 1: Simplified Microwave

Design and implement a class that simulates a microwave control panel with the following four buttons:

1. One for increasing the time by 30 seconds.
2. One for switching between power levels 1 and 2.
3. A start button.
4. A reset button.

In your Microwave class, each button is implemented as a method. The method for the start button should print a message “Cooking for … seconds at level …”.

Please also implement a main function that tests out all four buttons in meaningful combination and display clear messages to the user on what operation is carried out by the microwave.

|  |
| --- |
| **Your code for this problem** |
| **1 | package edu.banks.csc272;**  2 | ​  3 | /\*\*  4 |  \* Microwave class: Simulates a microwave control panel.  5 |  \*  6 |  \* Concepts Covered:  7 |  \* - Encapsulation (private fields)  8 |  \* - Methods as Behaviors (each button is a method)  9 |  \* - Constructor usage  10|  \* - Mutator and Accessor Patterns  11|  \* - Test code in main method  12|  \* (Lecture 1, CSC272)  13|  \*/  14| public class Microwave {  15|     // [Lecture 1: Encapsulation]  16|     private int time;         // in seconds  17|     private int powerLevel;   // 1 or 2  18| ​  19|     /\*\*  20|      \* [Lecture 1: Constructor]  21|      \* Default constructor: sets time to 0, power level to 1.  22|      \*/  23|     public Microwave() {  24|         time = 0;  25|         powerLevel = 1;  26|    }  27| ​  28|     /\*\*  29|      \* [Lecture 1: Mutator]  30|      \* Increases time by 30 seconds.  31|      \*/  32|     public void increaseTime() {  33|         time += 30;  34|         System.out.println("Time increased by 30 seconds. Total time: " + time + " seconds.");  35|    }  36| ​  37|     /\*\*  38|      \* [Lecture 1: Mutator & State Toggle]  39|      \* Switches between power level 1 and 2.  40|      \*/  41|     public void switchPowerLevel() {  42|         powerLevel = (powerLevel == 1) ? 2 : 1;  43|         System.out.println("Power level switched. Current level: " + powerLevel);  44|    }  45| ​  46|     /\*\*  47|      \* [Lecture 1: Behavior/Action]  48|      \* Starts the microwave, outputs settings.  49|      \*/  50|     public void start() {  51|         System.out.println("Cooking for " + time + " seconds at level " + powerLevel + ".");  52|    }  53| ​  54|     /\*\*  55|      \* [Lecture 1: Mutator & Reset Pattern]  56|      \* Resets the microwave to default settings.  57|      \*/  58|     public void reset() {  59|         time = 0;  60|         powerLevel = 1;  61|         System.out.println("Microwave reset to default settings.");  62|    }  63| ​  64|     /\*\*  65|      \* [Lecture 1: Testing & Demonstration]  66|      \* Main method tests all behaviors.  67|      \*/  68|     public static void main(String[] args) {  69|         Microwave mw = new Microwave();  70|         mw.increaseTime();  71|         mw.increaseTime();  72|         mw.switchPowerLevel();  73|         mw.start();  74|         mw.reset();  75|         mw.start();  76|    }  77| }  78| ​ |

Run the code and insert the result in the following box.

|  |
| --- |
| **Sample Run Result** |
| **/home/human/.jdks/ms-17.0.15/bin/java -javaagent:/app/extra/idea-IU/lib/idea\_rt.jar=43295 -Dfile.encoding=UTF-8 -classpath /home/human/Documents/GitHub/csc272-module1-homework/target/classes edu.banks.csc272.Microwave**  Time increased by 30 seconds. Total time: 30 seconds.  Time increased by 30 seconds. Total time: 60 seconds.  Power level switched. Current level: 2  Cooking for 60 seconds at level 2.  Microwave reset to default settings.  Cooking for 0 seconds at level 1.  ​  Process finished with exit code 0 |

# Problem 2: Student and Instructor

Design a superclass Person, and make two subclasses, Student and Instructor, that inherit from Person.

A person has a name and a year of birth. A student has a major. An instructor has a salary.

Implement class declarations, the constructors, and the methods toString for all classes.

Implement a test program (a separate class file) that tests these classes and methods.

|  |
| --- |
| **Your code for this problem** |
| **Person.java** package edu.banks.csc272;  ​  /\*\*  \* Represents a generic person with a name and year of birth.  \*  \* Concepts Covered:  \* - Encapsulation (private fields)  \* - Base class for inheritance [Lecture 2]  \* - Constructor pattern  \* - Overriding toString()  \*/  public class Person {     // [Lecture 2: Encapsulation & Inheritance]     private String name;     private int yearOfBirth;  ​     // [Lecture 2: Constructor]     public Person(String name, int yearOfBirth) {         this.name = name;         this.yearOfBirth = yearOfBirth;    }  ​     // [Lecture 2: toString() method for representation]     public String toString() {         return "Person[name=" + name + ", yearOfBirth=" + yearOfBirth + "]";    }  }    **Student.java package edu.banks.csc272;**  ​  /\*\*  \* Student is a subclass of Person with a major.  \*  \* Concepts Covered:  \* - Inheritance with 'extends' [Lecture 2]  \* - Additional fields in subclass  \* - Calling superclass constructor with super()  \* - Overriding toString()  \*/  public class Student extends Person {     // [Lecture 2: Subclass adds field]     private String major;  ​     // [Lecture 2: Constructor chaining with super()]     public Student(String name, int yearOfBirth, String major) {         super(name, yearOfBirth);         this.major = major;    }  ​     // [Lecture 2: Method overriding]     @Override     public String toString() {         return "Student[" + super.toString() + ", major=" + major + "]";    }  }  **Instructor.java**  **package edu.banks.csc272;**  ​  /\*\*  \* Instructor is a subclass of Person with a salary.  \*  \* Concepts Covered:  \* - Inheritance with 'extends' [Lecture 2]  \* - Additional field in subclass  \* - Calling superclass constructor with super()  \* - Overriding toString()  \*/  public class Instructor extends Person {     private double salary;  ​     public Instructor(String name, int yearOfBirth, double salary) {         super(name, yearOfBirth);         this.salary = salary;    }  ​     @Override     public String toString() {         return "Instructor[" + super.toString() + ", salary=" + salary + "]";    }  }  **PersonTester.java**  **package edu.banks.csc272;**  ​  /\*\*  \* Tests the Person, Student, and Instructor classes.  \*  \* Concepts Covered:  \* - Polymorphism (all can be treated as Person)  \* - Object instantiation  \* - Demonstrating method overriding  \*/  public class PersonTester {     public static void main(String[] args) {         Person p = new Person("Alice", 1980);         Student s = new Student("Bob", 2000, "Computer Science");         Instructor i = new Instructor("Carol", 1975, 85000);  ​         // [Lecture 2: Polymorphic toString() behavior]         System.out.println(p);         System.out.println(s);         System.out.println(i);    }  }  ​ |

Run the code and insert the result in the following box.

|  |
| --- |
| **Sample Run Result** |
| **/home/human/.jdks/ms-17.0.15/bin/java -javaagent:/app/extra/idea-IU/lib/idea\_rt.jar=40911 -Dfile.encoding=UTF-8 -classpath /home/human/Documents/GitHub/csc272-module1-homework/target/classes edu.banks.csc272.PersonTester**  Person[name=Alice, yearOfBirth=1980]  Student[Person[name=Bob, yearOfBirth=2000], major=Computer Science]  Instructor[Person[name=Carol, yearOfBirth=1975], salary=85000.0]  ​  Process finished with exit code 0 |

**The end**