CS390 – FPP MID EXAM-REVIEW

Midterm – Week-2-Saturday – 9.30 – 12.15 PM

Course Resources : www.online.cs.miu.edu

Reading Resources : Demo Code and Homework problems

Lessons for Examination

Lesson – 3 - Objects and Classes
Lesson – 4 - Recursion
Lesson – 5 – Inheritance, Interface, and Polymorphism
Lesson – 6 – Inner class(Sorting only)

Important Information

1. **Reporting Time & Venue**
   - Report by **9:30 AM** at **V17 classroom** for preliminary checks and IDE setup.
   - Exam runs from **9:30 AM – 12:15 PM**.
2. **Exam Mode**
   - You will take the exam on your own machine using **Sakai ProctorTrack**.
   - **Steps to follow:**
     a. Log in to Sakai → CS390 Course Site → ProctorTrack.
     b. Select the Midterm and click **Go to Test**.
     c. Read instructions carefully. Open **ProblemRequirements.pdf**.
     d. Download, extract the problem set, and copy the prob1, prob2, prob3 packages into your IDE workspace.
     e. Complete all tasks as described in the PDF.
     f. Do not alter startup code; only finish the requested tasks.
     g. Ensure your code compiles and runs without errors.
     h. On completion, compress your project folder and upload it to Sakai.
3. **After Submission**
   - Meet me at my table to run your code.
   - Verify your file is uploaded on Sakai. A copy will also be saved to my flash drive.
4. **Rules & Restrictions**
   - **Closed-book exam** – No materials allowed.

- **AI Assistant:** Disable auto-coding. Only basic context suggestions are allowed. AI usage will result in an **NP grade**.
- **Mobile & Devices:** Switch off/silent. No smartwatches or devices. Keep all belongings in a backpack at the front.
- **Academic Honesty:** Cheating results in **NP grade**.
- **Breaks:** Avoid restroom breaks unless urgent.

5. **Exam Content**
   - **Problem 1** – Recursion
   - **Problem 2** – Inheritance & Polymorphism(Regular/Abstract), Sorting
   - **Problem 3** – Interfaces & Polymorphism, Sorting

Problem Requirements

Recursion

- Write recursive methods with **base** and **recursive cases**.
- Able to work with numbers, strings, and arrays.
- Refer to examples from class demos and labs: strings, search algorithms, and math problems.

Inheritance, Interfaces & Polymorphism

- Inheritance & Polymorphism
- Abstract Classes & Polymorphism
- Interfaces & Polymorphism
- Converting non-OO code to OO-code
- Creating Immutable Classes

Inner Class (Sorting)

- Perform sorting on collections using **Comparator** with Inner Class concepts.
- Practice at least one approach:
  - Separate Class, Member Inner Class, Local Inner Class, Anonymous Inner Class, or Lambda.
