**Howard University**

**College of Engineering and Architecture**

**Department of Electrical Engineering & Computer Science**

**Large Scale Programming**

**Fall 2025**

**Midterm Exam**

October 20, 2025

## ****Instructions****

* **Exam Format:**  
  Your examination consists of both **essay** and **programming** problems.
* **Essay Questions:**  
  Complete all essay (rationale) questions **inline in this document**.  
  My preference is **Microsoft Word (.docx)**, but **.txt** or **.pdf** are acceptable alternatives.  
  Upload your completed essay file to your repository under the package:

org.howard.edu.lsp.midterm.doc

You may use **any file name**.

* **Programming Problems:**  
  Each programming problem must be uploaded to your repository using the package specified in the question.  
  For example:

org.howard.edu.lsp.midterm.question1

* **Committing Your Work:**  
  If using a third-party IDE or tool to commit, commit early and often.  
  Do not wait until the end of the exam to push your code.  
  If you encounter problems committing, you may manually upload your code to your repository.  
  If you are unable to commit or upload, you may zip your project and email it to  
  bwoolfolk@whiteboardfederal.com.  
  ⚠️ This will result in a 20% deduction from your final exam score.
* **Citations and References:**  
  You must cite all references for any material obtained from the internet.  
  Any AI-generated content (e.g., ChatGPT conversations) must be included in full.  
  Each package you upload must include a references document corresponding to that package’s content.  
  ⚠️ Failure to provide references will result in a zero for that question.
* **Exam Policy:**  
  This is an OPEN BOOK, OPEN NOTES exam.  
  Collaboration of any kind is strictly prohibited. Any violations will be handled in accordance with **university academic integrity guidelines**.

**Question 1. (20 pts.)**

**Given the following, analyze the class below and answer the below questions. This question does NOT require you to write any code.**

package org.howard.edu.lsp.studentPortalHelper;

import java.io.\*;

import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

import java.util.\*;

public class StudentPortalHelper {

// Data cache (in-memory)

private static final Map<String, String> cache = new HashMap<>();

// GPA calculation

public static double computeGPA(List<Integer> grades) {

if (grades == null || grades.isEmpty()) return 0.0;

int sum = 0;

int count = 0;

for (int g : grades) { sum += g; count++; }

double avg = (double) sum / count;

// simple mapping: 90–100=A=4, 80–89=B=3, etc.

if (avg >= 90) return 4.0;

if (avg >= 80) return 3.0;

if (avg >= 70) return 2.0;

if (avg >= 60) return 1.0;

return 0.0;

}

// CSV export to disk

public static void exportRosterToCsv(String filename, List<String> names) {

try (PrintWriter pw = new PrintWriter(new FileWriter(filename))) {

pw.println("name");

for (String n : names) {

pw.println(n);

}

} catch (IOException e) {

System.err.println("Failed to export roster: " + e.getMessage());

}

}

// Email formatting

public static String makeWelcomeEmail(String studentName) {

return "Welcome " + studentName + "! Please visit the portal to update your profile.";

}

// Date formatting (UI concern)

public static String formatDateForUi(LocalDate date) {

return date.format(DateTimeFormatter.ofPattern("MM/dd/yyyy"));

}

// Payment processing (stub)

public static boolean processTuitionPayment(String studentId, double amount) {

if (amount <= 0) return false;

// pretend to call external gateway...

return true;

}

// Password strength check (security)

public static boolean isStrongPassword(String pwd) {

if (pwd == null || pwd.length() < 8) return false;

boolean hasDigit = false, hasUpper = false;

for (char c : pwd.toCharArray()) {

if (Character.isDigit(c)) hasDigit = true;

if (Character.isUpperCase(c)) hasUpper = true;

}

return hasDigit && hasUpper;

}

// Ad-hoc caching

public static void putCache(String key, String value) {

cache.put(key, value);

}

public static String getCache(String key) {

return cache.get(key);

}

}

**Tasks:**

Using one or more **Arthur Riel heuristics**, analyze whether the StudentPortalHelper class demonstrates **high** or **low cohesion**.  
a) Should a well-designed class have high or low cohesion? Explain and defend your answer. (5 pts.)

**A well-designed class should have high cohesion. That means all of its methods and data focus on one clear purpose. High cohesion makes code easier to understand, and test. Low cohesion happens when a class mixes unrelated jobs which makes it confusing and harder to update without breaking other parts.**

b) Based on your analysis, discuss—**only if you believe changes are needed**—how you would reorganize or redesign the class to improve its structure. Your answer should (1) identify the class as having high, low or perfect cohesion and (2) describe a **general approach** to refactoring the class. If you believe the class already has good cohesion, justify why no changes are necessary. (15 pts)

*(If you believe the class already has good cohesion, justify why no changes are necessary.)*

**The Cohesion level for this class is low because it mixes too many responsibilities. These tasks don’t even relate to one another or share data. The best method to improve this class is to break it down into smaller classes where each handles its own job. Such as a GradeCalculator for GPA logic**

**Question 3.**

**Given the following, answer the below questions.**

**(20 pts.)**

**Given:**

A car manufacturer uses Java software to track current vehicles being built. The UML diagram below shows an excerpt of the current software structure. You should assume the presence of other appropriate fields and methods

**Car**

getTrimLevel()

**Engine**

accelerate()

getFuelLevel()

**Base**

**Sports**

**Luxury**

**Electric**

**Petrol**

Each car can be built to one of three trim levels: Base, Luxury or Sport. They can also be configured with an electric or petrol engine. At various points in the manufacturing process the customer can choose to change the trim level.

**Task:**

1. Explain in detail why the current structure does or does not support this. (10 pts.)

* **No. Trim level is modeled by inheritance (Base, Luxury, Sport extend Car).**  
   **In Java you can’t change an object’s class at runtime, so a Base car can’t “become” Luxury without creating a new instance and copying state. Which isn’t practical making this structure rigid.**

1. Describe how to refactor the structure to allow trim-level change for a car to dynamically change. Hint: How would you modify Car to use composition to solve the problem? (10 pts.)

* **To make trim changes possible, the class should use composition instead of inheritance. Instead of making Base, Luxury, and Sport subclasses of Car, we can give Car a Trim object as one of its fields. This allows the trim to be changed at any time using a setter method, like setTrim(new LuxuryTrim()). Each trim type can be a separate class that implements a Trim interface. This design makes the system flexible cars can switch trims dynamically without recreating the whole object.**

**Question 5 (10 pts)**  
**Reflection on AI Use in Learning and Problem Solving**

Discuss your personal experience using **AI tools** (such as ChatGPT, GitHub Copilot, or others) before and during this course.  
In your response, address the following points:

1. How have you used AI to support your learning or programming in this course?
2. What benefits or limitations did you encounter?
3. Looking ahead, how do you expect AI to influence the way you solve problems **academically or professionally**?

Your answer should be **1–2 well-developed paragraphs.**

**Before this course, I mainly used AI tools like ChatGPT to help me understand programming concepts and debug small coding errors. During this course, I began using AI more intentionally—to review syntax, get explanations for Java features like inheritance and polymorphism, and check the logic of my programs before testing them. It helped me quickly identify mistakes and understand why something wasn’t working, rather than just giving me the answer. I also used AI to practice explaining code in plain language, which made studying for quizzes and writing reports easier.**

**The main benefit I experienced was efficiency I could learn new material faster and get feedback immediately. However, I also learned that AI can be limited if I rely on it too much; sometimes, it gives code that compiles but doesn’t fit the assignment’s exact requirements. Looking ahead, I expect AI to become a regular part of how I solve problems in both school and professional settings. It will help me brainstorm, debug, and learn new technologies faster, but I’ll still need to apply critical thinking and verify results myself.**