**Howard University**

**College of Engineering and Architecture**

**Department of Electrical Engineering & Computer Science**

**Large Scale Programming**

**Fall 2024**

**Midterm Exam – Part I**

October 17, 2024

**Part I: 50 pts. (essay questions)**

**Instructions:**

1. **Submit completed exam to your github repository under package org.howard.edu.lsp.midterm**

**Verify that the commit(s) completed successfully**

1. **OPEN BOOK, OPEN NOTES. THERE IS NO COLLABORATION ON THIS EXAM**
2. **Site any and ALL references for anything obtained off the internet.**

Latanya Khissy Beyniouah

**Question 1. (10 pts.)**

You are reviewing the following class in a **Fitness Tracking System**. The class is intended to manage various aspects of a user’s fitness activities.



**Tasks:**

1. Analyze the **UserFitness** class. Explain why this class has **low cohesion.** (5 pts.)

* **This class has low cohesion because it is handling various responsibilities that are not related with one class. For example, the daily workout class is tracking steps, calories burned, and the type of workout done, which are unrelated and should be split into more focused classes. [**[used definition of low cohesion from geeksforgeeks](https://www.geeksforgeeks.org/software-engineering-coupling-and-cohesion/)**]**

1. Describe, in general, how you would split the **UserFitness** class into **more cohesive classes**. Your answer should not be a complete design. (5 pts.)

* **I would do this by grouping the main responsibilities into separate classes. Here are the classes I would create:**
  + **Storing user’s profile information**
    - **Name, age, weight, etc.**
  + **Tracking individual workouts**
  + **Tracking calories/meals**
* **I believe doing this would make it easier to update information without affecting other classes that are not related to one another.**

**Question 2. (5 pts.)**

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

A university manages its students using a program that has a class Student with subclasses FirstYear, SecondYear, and ThirdYear for year-specific state and behaviors. The program has a List that contains all Students.

Should Student be a class, an abstract class or an interface? Explain your answer

* **Student should be a class because it is a parent class to FirstYear, SecondYear, and ThirdYear. All of the subclasses inherit the behaviors from the Student class.**

1. At the end of each year, the third-year students graduate and must be removed. This is done by passing the list to the following method:

void removeThirdYears(List<Students> students) {

for (Student student : students) {

try {

ThirdYear = (ThirdYear) student;

students.remove(thirdyear);

}

}

}

Why is the above method flawed? Hint: Another Arthur Riel heuristic (5 pts.)

* **It’s flawed because the given list of students is being iterated over AND modified concurrently.** [based off of OO heuristics slides**]**

**Question 3. (5 pts.)**

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

public class BankAccount {

public String accountHolderName;

public String accountNumber;

public double balance;

public double interestRate;

public BankAccount(

String accountHolderName,

String accountNumber,

double initialDeposit,

double interestRate) {

this.accountHolderName = accountHolderName;

this.accountNumber = accountNumber;

this.balance = initialDeposit;

this.interestRate = interestRate;

}

// Exposed implementation for adding interest

public void addInterest() {

double interest = balance \* (interestRate / 100);

balance += interest;

}

// Exposed implementation for deposits

public void deposit(double amount) {

balance += amount;

}

// Exposed implementation for withdrawals

public void withdraw(double amount) {

balance -= amount;

}

}

1. Is BankAccount well-encapsulated? Why or why not? If so, describe what changes you would make to the class. Otherwise, state nothing. (5 pts.)

* **It is not well-encapsulated because all the information is public. I would follow object-oriented programming by making sensitive data private, like the accountNumber and balance to protect the sensitive information. [**slides**]**

**Question 4. (10 pts.)**

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

**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 not support this. (5 pts.)

* Based off of the OOP principles, the current structure doesn’t support this because of its inheritance hierarchy, lack of polymorphism, and static relationship.
  + Since the car class has a direct relationship with base, luxury, and sports classes, that implies it can only belong to one trim level at a time.
  + The method “getTrimLevel()” only returns the current trim and not the things needed to modify it.
  + Lastly, Car and its subclasses relationship is static because the trim level is determined when the object is created and can’t be modified.

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? (5 pts.)

* **To do this, we can change the Car class to have a TrimLevel property instead of making it a type of trim level. This would allow the Car class to be changed to a different trim level without having to create a new car. [**read through crc card slides**]**

**Question 5. (5 pts.)**

Suppose you are about to code a particular class and find that you have written another class with nearly the functionality that you now are developing. This means that the two classes in large parts contain the same code, but in smaller parts have different code. Explain in detail why this may be a problem and how to solve it. Give your answer in terms of Arthur Riel’s design heuristics.

* **Based off of the OO design heuristics slides, a potential problem with this is the duplication of code, which can lead to many maintenance and development issues. Having multiple classes share similar code would make the programmer have to apply any changes/fixes to all instances.**
* **I would solve this by:**
  + **Decreasing coupling**
  + **Increasing cohesion**
  + **Reusing existing code.**

**Question 6. (5 pts.)**

Suppose we need to store instances of the classes Car and BankAccount in a database. To obtain low coupling it may seem as if the functionality to save an instance should be in precisely those classes, since they have the information that shall be saved. Why is it unwise to place the database calls there?

* **This is unwise because it is putting too much responsibility on a single class. Doing this would make the information within the database harder to understand and more difficult to maintain.**

**Question 7. (5 pts.)**

Given a set of requirements, what conditions must be met to make it appropriate to create a new class hierarchy using inheritance? Neither superclass nor subclass exist before, the question is about designing a new class hierarchy.

* **Shared Characteristics**
* **Is-A Relationship**
* **Code Reusability**
* **Extensibility**
* **Behavioral Consistency**
* **Avoid Overengineering**
* **Domain Appropriateness**

**[used slides +** [**link**](https://codinginthetrenches.com/2016/02/20/class-hierarchy-design/)**]**

**Question 8. (2.5 pts.)**

Describe the difference between inheritance and composition why would you choose one over the other.

**Question 9. (2.5 pts.)**

Explain the meaning of cohesion and describe whether low or high cohesion is better.

* **Cohesion is how closely related a modules responsibilities are. High cohesion is better because it makes the code easier to understand and update since the functions are all in one place.**