**Howard University**

**College of Engineering and Architecture**

**Department of Electrical Engineering & Computer Science**

**Large Scale / Object-Oriented Programming**

**Midterm Exam – Part I**

March 14, 2024

**Name : Samuel Kusi**

**This exam is two parts:**

**Part I: 70 pts. (T/F, multiple choice, essays)**

**Part II: 30 pts. (programming)**

**Instructions:**

1. **Please return this exam in Word format, do not convert to pdf.**
2. **Submit completed exam to your github repository. Create package:**

**org.howard.edu.lsp.midterm**

**Verify that the commit completed successfully**

1. **OPEN BOOK, OPEN NOTES. THERE IS NO COLLABORATION ON THIS EXAM**

**Section 1: True/False. Highlight or write your answer. Each question is 1 pt.**

1. **T** The ability to hide the implementation details of an object is called encapsulation
2. **T** Java code is a compiled language that creates binary code for a specific architecture and must be recompiled when moved to another machine
3. **F** Polymorphism works in Java because method calls are bound to their method definitions at compile time
4. **T** Every object in Java explicitly derives from class Object using extends.
5. **F** A class in Java may implement only one interface
6. **T** In Java, when an instance of a class, or object, is specified as a parameter to a method, a reference to the said object is passed to the method
7. **F** When designing a class, conceptually, each class should be designed to have multiple goals so that your overall design can have less classes
8. **F** In Java, it is impossible to have memory leaks because of garbage collection
9. **F** A Java class can extend from multiple classes
10. **T** Testing does not ensure that our programs will never have any bugs
11. **F** According to principles of encapsulation, instance variables should always be declared as public
12. **F** Google naming conventions recommend that a Java class should begin with a lowercase letter and methods with an uppercase letter
13. **T** A subclass can add behavior that is not present in the superclass.
14. **T** If B is a subclass of A, then a B object may always be assigned to a variable of type A
15. **T** If class A extends class B, class A is a subclass of B and B is a superclass of A
16. **F** A constructor can be invoked multiple times once an object is instantiated
17. **T** A checked exception represents an error that a program has to handle
18. **F** In Java, the **this** keyword can be used to refer to both the current instance of the class and other instances of the class within the same scope
19. **F** Methods are said to be *overridden* if they are in the same scope and have the same names but different signatures
20. **T** A method that *overrides* another must have the same name but a different signature
21. **F** A static method can refer to any instance variable of the class
22. **F** Refactoring code involves changing the functionality of the code without modifying its external behavior
23. **T** A Java method returns void if it does not return a value
24. **F** An ArrayList in Java will not automatically change its size as needed.
25. **F** Polymorphism works in Java because method calls are bound to their method definitions at compile time.

**Section 2: Multiple Choice, type answer below each question. Each question is 1 pt.**

1. Which of the following is also known as run-time binding or late binding?

Dynamic binding

1. Which principle suggests that each module should perform a single, well-defined task?

High Cohesion

1. A good design should strive for:

a tight cohesion and a loose coupling between modules

1. Which of the following is true about composition in object-oriented programming?

It is a way of creating complex objects by combining simpler ones.

1. In Java, when can the garbage collector collect the memory allocated to an object?

when it can prove that there is no reference to that object from any other object

1. What is the purpose of the Single Responsibility Principle (SRP) in object-oriented design?

To ensure that each class has only one responsibility and reason to change

1. What is an abstract class?

A class that has no direct instances, but whose descendants may have direct instances

1. What is true about “has-a” and “is-a” relationships? **(Choose two)**

instance variables can be used when creating a has-a relationship

inheritance represents an is-a relationship

1. How does Arthur Riel's principle of "information hiding" contribute to better software design?

By encapsulating the implementation details of a module

1. According to Arthur Riel's design principles, what should be the relationship between modules in a well-designed system?

Modules should be loosely coupled to each other

1. Which principle suggests that each module should perform a single, well-defined task?

High Cohesion

1. What principle suggests that subclasses should be substitutable for their base classes without affecting the correctness of the program?

Liskov Substitution Principle

1. What is the primary purpose of inheritance in object-oriented programming according to Arthur Riel's principles?

To promote code reuse and minimize redundancy

1. Given the following. What is true?

public class Room {

public int roomNr;

private Date beginDtm;

private Date endDttm;

public void book(int roomNr, Date beginDttm, Date endDttm) {

this.roomNr = roomNr;  
 this.beginDtm = beginDttm;  
 this.endDttm = endDttm;

}

}

the variable roomNr breaks encapsulation

1. What can directly access and change the value of the variable roomNr?

package com.mycompany;

public class Hotel {

protected int roomNr = 100;

}

any class in com.mycompany package

1. What is the output of the following?

public class Example {

public static void main(String[] args) {

String str1 = "hello";

String str2 = new String("hello");

System.out.println(str1 == str2);

}

}

false

1. Which of the following statements is incorrect in Java?

private members of a class can be inherited by a sub class, and become protected members in sub class

1. Which of the following allow us to define an IS-A relationship in Java? Circle **all** that apply.

interfaces

classes

1. Which of these can be overloaded? (choose one or more answers)

methods

constructors

1. Given:

public interface Jumper { public void jump(); }

public class Animal { }

public class Dog extends Animal { protected Tail tail; }

public class Beagle extends Dog implements Jumper {

public void jump() {};

}

public class Cat implements Jumper { public void jump(); }

Which of the following are true?

Cat is-a Jumper

Dog is-a Animal

Beagle has-a Tail

1. Which of the following statements is true about checked exceptions in Java?

They must be caught or declared in the method signature using the “throws” clause

1. What happens if an exception is thrown within a “try” block but is not caught by any “catch” block?

The program crashes with a runtime error

1. A team manager needs data about the changes that have taken place for AWS resources in his account during the past two weeks. Which AWS service can help get this data?

AWS Cloud Trail

1. A company wants to establish a private, dedicated connection between AWS and its on-premises data center. Which AWS service is the right choice for this requirement?

AWS Direct Connect

1. Which of the following will help you control the incoming traffic to an Amazon EC2 instance?

Security Group

1. Which of the following statements is true about interfaces in Java?

A class can implement multiple interfaces

1. A company is looking at real-time processing of streaming big data for their ad-tech platform. Which of the following AWS services is the right choice for this requirement?

Amazon Kinesis Data Streams

1. An e-commerce application sends out messages to a downstream application whenever an order is created. The downstream application processes the messages and updates its own systems. Currently, the two applications directly communicate with each other. Which service will you use to decouple this architecture, without any communication loss between the two systems?

Amazon Simple Queue Service (SQS)

1. Which of the following AWS services will help provision a logically isolated network for your AWS resources?

Amazon Virtual Private Cloud (Amazon VPC)

1. Which of the following are the security best practices suggested by AWS for Identity and Access Management (IAM)? (Select two)

Do not share security credentials between accounts, use IAM roles instead

When you create IAM policies, grant the least privileges required to perform a task

**Section 3: Short essay questions. Please limit your responses to a few sentences.**

* + - 1. You are writing an external service that a client program attempts to connect to. What type of exception (check or unchecked) do you believe we should use when the client program provides an invalid web address? Explain why.

**(5 pts.)**

For invalid web addresses provided by the client program, it's appropriate to use a checked exception. This ensures that the client program handles the exception explicitly, prompting the user to correct the input or take appropriate action.

* + - 1. Java programs are said to be write once, run anywhere. Please explain this statement.

**(5 pts.)**

The statement "Java programs are write once, run anywhere" refers to Java's platform independence. Java code, once written, can be compiled into bytecode that runs on any device with a Java Virtual Machine (JVM), regardless of the underlying hardware or operating system.

* + - 1. **Explain the difference between abstract classes and interfaces in object-oriented programming. Discuss when it is appropriate to use each construct and the implications for code design.**

**(5 pts.)**

Abstract classes and interfaces are both tools for abstraction in object-oriented programming, but they serve different purposes:

Abstract classes: These act as blueprints for subclasses, defining a template for shared attributes and behaviors. They can have concrete (implemented) methods that subclasses inherit, along with abstract methods that subclasses must implement. Use abstract classes when you want to enforce a common structure and provide some default functionality for related classes.

Interfaces: These define contracts that a class must adhere to. They only specify method signatures (what the method does) without any implementation details (how it does it). Classes can implement multiple interfaces. Use interfaces to promote loose coupling and enforce specific functionalities that can be shared by unrelated classes.