**Howard University**

**College of Engineering and Architecture**

**Department of Electrical Engineering & Computer Science**

**Large Scale Programming**

**Final Exam – Part I**

December 6, 2024

**Name :\_\_\_Oluwayimika Adeyemi\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**This exam is two parts:**

**Part I: 50 pts. (T/F, multiple choice)**

**Part II: 50 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.finalexam**

**Verify that the commit completed successfully. Feel free to also e-mail it to bwoolfolk@whiteboardfederal.com**

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 FInheritance provides a mechanism by which changes to lower-level classes can be propagated to all super classes quickly.
2. T F In Java, the signature of a method is completely specified by the name of the method only.
3. TF The ability to hide the implementation details of an object is called polymorphism.  
   Resource: https://www.simplilearn.com/tutorials/java-tutorial/java-encapsulation#:~:text=Encapsulation%20in%20Java%20refers%20to,in%20which%20they%20are%20found.
4. T F Design patterns are a mechanism that enable developers to reuse code in their implementations.
5. TF Polymorphism works in Java because method calls are bound to their method definitions at compile time.  
   Resource: https://www.geeksforgeeks.org/compile-time-polymorphism-in-java/
6. T FFactory pattern can be combined with other patterns.
7. TFEvery object in Java **explicitly** derives from class Object using extends.
8. TF A class in Java may implement multiple interfaces.
9. TFTesting ensures that our programs will never have any bugs.
10. TF Google naming conventions recommend that a Java class should begin with a lowercase letter and methods with an uppercase letter  
    Resource: https://google.github.io/styleguide/javaguide.html
11. T F A subclass can add behavior that is not present in the superclass.
12. T F If B is a subclass of A, then a B object may always be assigned to a variable of type A
13. TF If class A extends class B, class A is a subclass of B and B is a superclass of   
    Skipped
14. TF A unchecked exception represents an error that a program has to handle.
15. TF Methods are said to be *overridden* if they are in the same scope and have the same names but different signatures
16. TFA method that *overrides* another must have the same name but a different signature
17. TF A static method can refer to any instance variable of the class
18. TFAn ArrayList in Java will not automatically change its size as needed.  
    Resource: https://www.geeksforgeeks.org/arraylist-in-java/
19. T F The relationship between two objects related by composition cannot be changed at runtime.
20. T F When iterating a Java HashSet, you are guaranteed to retrieve objects stored in the same order they were inserted

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

1. Which of the following option leads to the portability and security of Java?
   1. Bytecode is executed by JVM
   2. Use of exception handling
   3. Dynamic binding between objects
   4. Proper encapsulation of classes and objects.
2. Which principle suggests that each module should perform a single, well-defined task?
   1. High Cohesion
   2. Low Coupling
   3. Stepwise Refinement
   4. Information Hiding  
        
      Resource: https://www.geeksforgeeks.org/cohesion-in-java/
3. Which of the following is true about composition in object-oriented programming?
   1. It is a form of inheritance.
   2. It allows objects to inherit properties and behaviors from another class.
   3. It is a way of creating complex objects by combining simpler ones.
   4. It is used to hide the implementation details of a class.
4. What is the purpose of the Single Responsibility Principle (SRP) in object-oriented design?
   1. To ensure that each class has only one method
   2. To ensure that each class has only one instance variable
   3. To ensure that each class has only one responsibility and reason to change
   4. To ensure that each class has only one constructor
5. What is the primary purpose of unit testing?
   1. To verify the correctness of the entire system.
   2. To test the integration between different components.
   3. To validate that individual units of code work as expected.
   4. To assess the performance of the application.
6. What is regression testing?
   1. Testing the system in various environments.
   2. Repeating previous tests to ensure existing functionality is not affected by changes.
   3. Testing the performance of the system under load.
   4. Verifying the correctness of a single unit of code.
7. What is an abstract class?
   1. A class that has direct instances, but whose descendants may have direct instances
   2. A class that has direct instances, but whose descendants may not have direct instances
   3. A class that has no direct instances, but whose descendants may have direct instances
   4. All of the mentioned
8. What is true about “has-a” and “is-a” relationships? **(Choose all that apply)**
9. instance variables can be used when creating a has-a relationship
10. inheritance represents an is-a relationship
11. inheritance represents a has-a relationship
12. instances must be used when creating a has-a relationship
13. How does Arthur Riel's principle of "information hiding" contribute to better software design?
    1. By minimizing the dependencies between modules
    2. By encapsulating the implementation details of a module
    3. By ensuring that each module has only one responsibility
    4. By maximizing cohesion within modules
14. What principle suggests that subclasses should be substitutable for their base classes without affecting the correctness of the program?
    1. Open/Closed Principle
    2. Liskov Substitution Principle
    3. Single Responsibility Principle
    4. Interface Segregation Principle  
         
       Resource: https://tusharghosh09006.medium.com/liskov-substitution-principle-lsp-744eceb29e8#:~:text=The%20Liskov%20Substitution%20Principle%20(LSP,the%20correctness%20of%20the%20program.
15. What is the primary purpose of inheritance in object-oriented programming according to Arthur Riel's principles?
    1. To increase the complexity of the software system
    2. To promote code reuse and minimize redundancy
    3. To encourage tight coupling between classes
    4. To decrease the number of classes in the system
16. Which of the following allow us to define an Is-A relationship in Java? Circle **all** that apply.
17. interfaces
18. classes
19. local variables
20. dynamic binding
21. none of the above
22. Given the following. What is true?

public class Room {

private 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;

}

}

1. the code demonstrates polymorphism
2. the class is fully encapsulated
3. the variable roomNr breaks encapsulation
4. variables roomNr, beginDttm and endDttm break polymorphism
5. the method book breaks encapsulation
6. What can directly access and change the value of the variable roomNr?

package com.mycompany;

public class Hotel {

protected int roomNr = 100;

}

* 1. only the Hotel class
  2. any class
  3. any class in com.mycompany package
  4. any class that extends Hotel

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);

}

}

* 1. true
  2. false
  3. Compilation Error
  4. Runtime Error

1. Which of these can be overloaded? (choose one or more answers)
   1. methods
   2. constructors
   3. classes
   4. interfaces
2. 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?

1. Cat is-a Animal
2. Cat is-a Jumper
3. Dog is-a Animal
4. Dog is-a Jumper
5. Cat has-a Animal
6. Beagle has-a Tail
7. Beagle has-a Jumper
8. Which of the following statements is true about checked exceptions in Java?
   1. They must be caught or declared in the method signature using the “throws” clause
   2. They are subclasses of RuntimeException
   3. They occur at runtime
   4. They do not need to be handled explicitly.
9. What happens if an exception is thrown within a “try” block but is not caught by any “catch” block?
   1. The program continues to execute normally
   2. The program crashes with a runtime error
   3. The “finally” block is executed
   4. The exception is automatically caught by the JVM
10. Which of the following is true about design patterns? (Choose the best answer).
11. Design patterns represent the best practices used by experienced object-oriented software developers.
12. Design patterns are solutions to general problems that software developers faced during software development.
13. Design patterns are obtained by trial and error by numerous software developers over quite a substantial period.
14. All of the above.
15. You want all the clients using class A to use the same instance of class A when the class is instantiated, what should you do to achieve this goal?
    1. Mark class A final
    2. Mark class A abstract
    3. Apply the Singleton pattern to class A
    4. Apply the Proxy pattern to class A
16. You have a class that accepts and returns values in British Imperial units (feet, miles, etc.), but you need to use metric units. The design pattern that would best solve your problem is:
17. Adapter
18. Decorator
19. Delegation
20. Proxy
21. Which of the following describes the Facade pattern correctly?
    1. This pattern allows a user to add new functionality to an existing object without altering its structure.
    2. This pattern is used when we need to treat a group of objects in a similar way as a single object.
    3. This pattern hides the complexities of the system and providers an interface to the client using which the client can access the system.
    4. This pattern is primarily used to reduce the number of objects created and to decrease memory footprint and increase performance.
22. Which of the following are concerned with communication between objects?
    1. J2EE Design Patterns
    2. Behavioral Design Patterns
    3. Structural Design Patterns
    4. Creational Design Patterns
23. What is the role of the Template Method in the Template Method design pattern?
    1. To ensure a class has only one instance and provides a global point of access to it.
    2. To define a family of algorithms, encapsulate each one, and make them interchangeable.
    3. To provide an interface for creating families of related or dependent objects without specifying their concreate classes.
    4. To define the skeleton of an algorithm in the superclass but let subclasses override specific steps of the algorithm without changing its structure.
24. Which of the following describes the Factory pattern correctly?
    1. This pattern creates an object without exposing the creation logic to the client and refers to newly created objects using a common interface.
    2. In this pattern, an interface is responsible for creating a factory of related objects without explicitly specifying their classes.
    3. This pattern involves a single class that is responsible to create an object while making sure that only a single object is created.
    4. This pattern is used when we want to pass data with multiple attributes in one shot from client to server.

Resource: https://refactoring.guru/design-patterns/factory-method

1. In the Command design pattern, what is the purpose of the Command interface?
   1. To provide an interface for creating families of related or dependent objects without specifying their concrete classes.
   2. To encapsulate a request as an object, thereby allowing for parameterization of clients with different requests.
   3. To define a family of algorithms, encapsulate each one, and make them interchangeable.
   4. To ensure a class has only one instance.
2. This design pattern should be used to access the contents of a collection without exposing its internal representation, to support multiple traversals of a collection, and to provide a uniform interface for traversing different collections.
   1. Template method
   2. Strategy
   3. Iterator
   4. Factory method
3. Which design pattern should you use when you want to provide a simple interface to a complex subsystem?
4. Adapter
5. Facade
6. Abstract Factory
7. Singleton
8. What is the intent of the Adapter design pattern?
   1. To provide an interface for creating families of related or dependent objects without specifying their concrete classes.
   2. To convert the interface of a class into another interface clients expect.
   3. To ensure a class has only one instance.
   4. To define a family of algorithms, encapsulate each one, and make them interchangeable.