**Howard University**

**College of Engineering and Architecture**

**Department of Electrical Engineering & Computer Science**

**Large Scale Programming, Fall 2022**

**First Exam**

October, 27, 2022

**Due: 11:59PM**

**Name : Brian Paul**

**Instructions:**

Your mid-term will be submitted via git.  In addition, once uploaded, everyone should send an e-mail to bwoolfolk@whiteboardfederal.com.  The content should be:

1. Send me your git URL and password (although you should be using the same repository).
2. Acknowledge that you did not collaborate with anyone on this exam.  **Failure to do this will result in me not accepting your exam**.

The package structure for your git repository should be:

**org.howard.edu.lsp.midterm.exam  (place your exam/word document here, or just e-mail it to me, whichever is easiest)**

**org.howard.edu.lsp.midterm.problem1 (Place all code for problem 1)**

**org.howard.edu.lsp.midterm.problem2 (Place all code for problem 2)**

**Section 1: True/False – mark answer in an appropriate manner, i.e., highlight in another color, write the answer, etc. (25 pts.)**

1. T F Java code is commonly referred to as “write-once run anywhere” because the .class files that the jvm produces can execute on many different hardware platforms.
2. T F Following the principles of data encapsulation, we should declare all instance variables to be public.
3. T F Google naming conventions recommend that a Java class should begin with a lowercase letter.
4. T F Google naming conventions recommend that a Java method should begin with an uppercase letter.
5. T F The ability to hide the implementation details of an object is called dynamic binding.
6. T F If B is a subclass of A, then a B object may always be assigned to a variable of type A.
7. T F If class A extends class B, class A is a subclass of B and B is a superclass of A.
8. T F Methods are said to be *overridden* if they are in the same scope and have the same names but different signatures.
9. T FA method that *overrides* another must have the same name but a different signature.
10. T F A Java class can extend from more than one class.
11. T F A Java class can implement as many interfaces as it sees fit.
12. T FAn array in Java will automatically change its size as needed.
13. T FIf a Java source file is compile successfully, the compilation process will produce one or more files with a .java extension.
14. T FThe Java Virtual Machine (JVM) is used to compile as opposed to execute Java programs.
15. T FThe relationship between two objects related by inheritance can be dynamically changed at runtime.
16. T F Neither an abstract class nor an interface can be instantiated as an object.
17. T FIn class inheritance, a subclass only inherits the non-private member of the super class.
18. T FTesting ensures that our programs will never have any bugs.
19. T F A constructor can be invoked multiple times once an object is instantiated.
20. T F Every class instantiated/created in Java is actually a sub-class of class Object.
21. T F If an exception gets thrown inside a try block, but none of the catch blocks specify the kind of exception that was thrown, then Java executes the rest of the try block.
22. T F An abstract method, gives a vague idea of the actual object being represented, but does not specify all the details..,
23. T F Many different classes may implement a single interface.
24. T F A checked exception represents an error that a program has to handle.
25. T F A runtime exception represents an error that a program has to handle.

**Section 2: Multiple Choice – mark answer in an appropriate manner, i.e., highlight in another color, write the answer, etc. (25 pts.)**

1. Which of the following is one of the benefits of object-oriented programming?
   1. enables code reuse
   2. more understandable code
   3. more maintainable code
   4. all of the above
2. The degree of interaction between two objects is called:
   1. polymorphism
   2. coupling
   3. inheritance
   4. instantiation
3. An object-oriented design tends to focus on blank to identify objects: (1pt.)
   1. verbs
   2. nouns
   3. encapsulation
   4. inheritance
4. Which of the following characteristics of an object-oriented programming language restricts behavior so that an object can only perform actions that are defined for its class?
   1. Dynamic Binding
   2. Polymorphism
   3. Inheritance
   4. Encapsulation
5. Which of the following statements about the Java Programming Language is true?
   1. Both procedural and OOP are supported
   2. Only procedural is supported
   3. Only OOP is supported
   4. All of the above
6. Which of the following is considered a blueprint that defines the methods and variables common to all objects of its specific kind
   1. Object
   2. Class
   3. Method
   4. Real data types
7. JUnit is used for what type of software testing for the Java language?

a) Functional Testing  
b) System Testing  
c) Unit Testing  
d) Integration Testing

1. JUnit test files are written in files with which file extension?
   1. .unit
   2. .java
   3. .test
   4. .junit
2. JUnit test methods must compulsorily return what value?

a) void  
b) Object  
c) String  
d) int

1. Which of the following test has to be completed before integration testing?

a) Unit testing  
b) Load testing  
c) Stress testing  
d) Functional testing

1. During a Java program run, what happens when the JVM tries to open an input file that does not exist?
   1. The JVM will throw a FileNotFoundException
   2. The JVM will treat it as if it was an empty input file.
   3. The JVM will skip over all further program statements relating to this file.
   4. The JVM will terminate the program without throwing an exception.
2. Which of the following is true about RuntimeException and its subclasses?
   1. If a method throws a RuntimeException, the use of the try/catch block is optional. \*\*\*\*\*\*
   2. The FileIOException class is a subclass of RuntimeException.
   3. In general, handling of RuntimeException should be done at compile time.
   4. In general, RuntimeException must be caught with a try/catch block.
3. The term signature can be used when describing a method. In this context, a signature is:

a) Defined by the import statement for that class.

b) The number of arguments and the data type of each argument.

c) The return type of method (for example, double or String).

d) The variable names in the argument list.

e) All of the above

1. Which of the following statements about constructors is correct:

a) A constructor has the same name as the class name.

b) A constructor is responsible for the initialization of an object's instance fields.

c) Constructor methods have no return type.

d) A class can have several constructors.

e) All of the above.

1. A class can have many methods with the same name as long as the number of parameters is different. This OOP concept is known as:
2. Method invocating
3. Inheritance
4. Method overriding
5. Method overloading
6. Given the code. 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;

}

}

1. The code demonstrates polymorphism
2. The class is fully encapsulated
3. The variable roomNr breaks encapsulation
4. Variables beginDttm and endDttm break polymorphism
5. The method book breaks encapsulation
6. Which can have access to private attributes of a class?
7. Only static methods of the same class
8. Only instances of the same class
9. Only methods defined in the same class
10. Only classes defined in the same package
11. Which of the following statements about Java access levels is FALSE?
    1. A private method can access a public field of another class.
    2. A private method can only be accessed in its own class.
    3. A private data field can be read but not modified from outside the class.
    4. A method without explicit access level is visible from the same p
12. Which of the following statements about Java class Object is TRUE? (hint: more than one answer)
13. It is a subclass of all other Java classes
14. It is the only class which has no parent class
15. It has a toString method which prints the name and value of all fields.
16. It has an equals method which compares objects by the value of their fields.
17. Which AWS networking service enables a company to create a virtual network within AWS?
18. AWS Config
19. Amazon Route 53
20. AWS Direct Connect
21. Amazon Virtual Private Cloud (Amazon VPC)
22. Which service would be used to send alerts based on Amazon CloudWatch alarms?
    1. Amazon Simple Notification Service (Amazon SNS)
    2. AWS CloudTrail
    3. AWS Trusted Advisor
    4. Amazon Route 53
23. Which AWS service would simplify the migration of a database to AWS?
    1. AWS Storage Gateway
    2. AWS Database Migration Service (AWS DMS)
    3. Amazon EC2
    4. Amazon AppStream 2.0
24. Why is AWS more economical than traditional data centers for applications with varying compute workloads?
    1. Amazon Elastic Compute Cloud (Amazon EC2) costs are billed on a monthly basis.
    2. Customers retain full administrative access to their Amazon EC2 instances.
    3. Amazon EC2 instances can be launched on-demand when needed.
    4. Customers can permanently run enough instances to handle peak workloads.
25. How would a system administrator add an additional layer of login security to a user's AWS Management Console?
26. Use Amazon Cloud Directory
27. Audit AWS Identity and Access Management (IAM) roles
28. Enable Multi-Factor Authentication
29. Enable AWS CloudTrail
30. The following statements compare and contrast array and ArrayList. Which one is true?
    1. Both ArrayList and array objects automatically keep track of their capacity and the number of elements actually in use.
    2. The array type extends from Object. The ArrayList type extends from array.
    3. The elements in an ArrayList can be primitives (int, float, double etc.) or reference-to values. An array can store only reference-to values.
    4. ArrayList can expand its storage space as needed; an array cannot change the initial size of the array.
    5. All of the above.

(section 4 on next page)

In a Java program, suppose that a certain entity has the name MESSAGE. Assuming standard Java

naming conventions, what kind of entity is this most likely?

[A] A constant

[B] A class

[C] A local variable with a changing value

[D] A package

In a Java program, suppose that a certain entity has the name MESSAGE. Assuming standard Java

naming conventions, what kind of entity is this most likely?

[A] A constant

[B] A class

[C] A local variable with a changing value

[D] A package

In a Java program, suppose that a certain entity has the name MESSAGE. Assuming standard Java

naming conventions, what kind of entity is this most likely?

[A] A constant

[B] A class

[C] A local variable with a changing value

[D] A package

In a Java program, suppose that a certain entity has the name MESSAGE. Assuming standard Java

naming conventions, what kind of entity is this most likely?

[A] A constant

[B] A class

[C] A local variable with a changing value

[D] A package

In a Java program, suppose that a certain entity has the name MESSAGE. Assuming standard Java

naming conventions, what kind of entity is this most likely?

[A] A constant

[B] A class

[C] A local variable with a changing value

[D] A package

In a Java program, suppose that a certain entity has the name MESSAGE. Assuming standard Java

naming conventions, what kind of entity is this most likely?

[A] A constant

[B] A class

[C] A local variable with a changing value

[D] A package

In a Java program, suppose that a certain entity has the name MESSAGE. Assuming standard Java

naming conventions, what kind of entity is this most likely?

[A] A constant

[B] A class

[C] A local variable with a changing value**Section 4: Progr**

**Section 4: Programming**

You are responsible for uploading your solution to your git repository. You should create a new parent package called **org.howard.edu.lsp.midterm**

**Problem #1:** **(30 pts.)**

Requirements:

Designing an electronic voting machine is a challenging task. You are asked to implement a very simplified voting machine class with the following requirements. A voting machine has a list of candidates and the following methods:

**package org.howard.edu.lsp.midterm.problem1;**

public class VotingMachine {

// YOU must decide upon an appropriate data representation to associate a name with respective votes. Create whatever you like.

public void addCandidate(String name) {

// Add a candidate to the list, initialize number of votes to 0

}

public void castVotes(String name, int votes) {

// Cast votes to the candidate with the given name

}

public int getVotes(String names) throws UnknownCandidateException {

// Return the number of votes for a given candidate. Throws

// UnknownCandidateException if name not present. Make this a

// checked exception.

}

public int sum() {

// Return the total number of votes for all candidates.

}

}

In summary, you are responsible for the following:

1. Implementation of the aforementioned methods. (5 pts. for each, 20 pts. total)
2. JUnit test cases for getVotes and sum methods. Use appropriate naming conventions for your test class and respective methods in it. (5 pts. for each, 10 pts. total)
3. Make sure you include javadocs in your source code but you do not have to generate html.

**Problem #2**: **(20 pts.)**

A programmer has created the following class Person to represent person records:

**package.org.howard.edu.lsp.midterm.problem2;**

public class Person {

private String name;

private int age;

private String socialSecurityNumber;

public Person(String name, String socialSecurityNumber, int age) {

// initialize private variables

}

}

In summary, you are responsible for the following:

* + - 1. Write an appropriate constructor for class Person. (5 pts.)
      2. In class Person, override the default implementation of the toString() method and return the person’s name, age and social security number. (5 pts.)
      3. In class Person, override the default implementation of the equals(Object) method. Two Person instances are equal if their social security numbers are the same. (10 pts.)

…

Person p1 = new Person(“name1”, “111-11-1111”);

Person p2 = new Person(“name2”, “111-11-1111”);

If (p1.equals(p2)) {

System.out.println(“These are the same!”); // in this case, they would be the same

} else {

System.out.println(“Not the same!”);

}

…

* + - 1. You can prove your implementation works by either providing an appropriate Driver or writing JUnit test cases. Your choice.