USN | | | | | | | | | | |

Internal Assessment Test I – Dec 2023

Sub:	OOPS WITH JAVA					Sub Code:	BCS306A	Branch:	CSE
Date:	18/01/24	Duration:	90 mins	Max Marks:	50	Sem/Sec:	III A,B,C	OBE	

Answer any FIVE FULL Questions

		MARKS	CO	RBT
1. a)	Explain how access modifiers (protected, default) influence the visibility of classes within and outside a package.	[5]	CO3	L2
1 b)	Describe how Java supports multiple inheritances through interfaces. Write a java program to achieve multiple inheritances using interface concept.	[5]	CO3	L2,L3
2 a)	Does the below Java code with abstract method compile? If yes provide implementation for method. If no write the reason and correct the error. class Puppy { abstract void showName();}	[5]	CO3	L3
2 b)	What is overloading. Write the difference between method overloading and constructor overloading. (Min 4 points).	[5]	CO3	L2

3 a)	What is the output of the below Java program with an abstract class?
	final abstract class Bell
	{ }
	class DoorBell extends Bell{
	DoorBell()
	{System.out.println("DoorBell ringing..");}}
	public class AbstractClassTesting2
	{public static void main(String[] args){
	Bell bell = new DoorBell();}}
		[5]	CO3	L3

3 b)	Write a java program to create one interface CreditCard with 2 methods accptRupees() and acceptDoller(). Provide implementation for both methods and print the output.	[5]	CO3	L3
4 a)	Explain the hierarchy of exceptions in Java. How are checked and unchecked exceptions related in the hierarchy with diagram?	[5]	CO4	L2

4 b)	What is the output of the below Java code?
	public class ExceptionTest5{
	public static void main(String[] args) {
	int ary[] = new int[2];
	ary[10] = 5;
	try { int number= 2/0;}
	catch(Exception e){ System.out.println("Divide by Zero"); }
	finally{System.out.println("Inside FINALLY block"); }}}
		[5]	CO3	L3

5 a)	How is super used to call the constructor of the super class? Write program to call super class constructor.	[5]	CO3	L3
5 b)	Write a java program to use static and default methods inside the interface and access them.	[5]	CO4	L3

PTO

6 a)	An object of multi-level inherited abstract class cannot be created in memory? State TRUE or FALSE.	[1]	CO3	L1
6 b)	Choose a correct statement about Java Interfaces? A) Interface contains only abstract methods by default. B) A Java class can implement multiple interfaces C) An Interface can extend or inherit another Interface. D) All the above	[1]	CO3	L1
6 c)	Which is the correct syntax to import a Java package below? A) import PACKAGE1.*; B) import PACKAGE1.CLASS1; C) import PACKAGE1.PACKAGE2.PACKAGE3.*; D) All the above	[1]	CO4	L1
6 d)	In java, can an abstract class be instantiated. Yes or no.	[1]	CO3	L1
6 e)	In abstract class we can write abstract methods and non abstract methods. True or false.	[1]	CO3	L1
6 f)	Which keyword used to declare a package?	[1]	CO4	L1
6 g)	Finally block is related to Exception in java. True or false.	[1]	CO4	L1
6 h)	Write syntax to call super class methods.	[1]	CO3	L1
6 i)	Write difference between Exception and Errors.	[1]	CO3, CO4	L1
6 j)	Method overriding is a compile time polymorphism. True/False	[1]	CO3	L1

CI **CCI** **HOD**

USN | | | | | | | | | | |

Internal Assessment Test II – Jan 2024

Sub:	OOPS WITH JAVA					Sub Code:	BCS306A	Branch:	CSE
Date:	18/1/24	Duration:	90 mins	Max Marks:	50	Sem/Sec:	III A,B,C		OBE

Answer any FIVE FULL Questions

		MARKS	CO	RBT

1. a)	Explain how access modifiers (protected, default) influence the visibility of classes within and outside a package.
	
	Explanation - In Java, access modifiers are keywords that control the visibility of classes, methods, and fields in a program. There are four access modifiers in Java: public, private, protected, and the default (package-private) modifier. These modifiers determine which other classes can access the members of a class and in what context.
	Here's how the **protected** and default (package-private) access modifiers influence the visibility of classes within and outside a package:
	Protected Access Modifier:
	If a class is declared with the protected modifier, it is accessible within its own package and by subclasses (regardless of whether they are in the same package or a different one). Members with protected access are also accessible within the same package and by subclasses, whether inside or outside the package.
	
	`package com.example;`
	`protected class ProtectedClass {`
	`  protected void protectedMethod() {`
	`  // accessible within the package and by subclasses`
	`  }`
	`}`
	Default (Package-Private) Access Modifier:
	If no access modifier is specified (default), it is also known as package-private. Classes with default access are accessible only within the same package.
	Members with default access are accessible only within the same package.
	`package com.example;`
	`class DefaultClass {`
	`  void defaultMethod() {`
	`  // accessible only within the package`
	`  }`
	`}`
		[5]	CO3	L2

1 (b)	Describe how Java supports multiple inheritances through interfaces. Write a java program to achieve multiple inheritances using interface concept.
	
	Solution:
	
	Write up Multiple Inheritance–
	Java supports multiple inheritances through interfaces, which are a way to achieve abstraction and allow a class to inherit the behaviors (method signatures) of multiple interfaces. In Java, a class can implement multiple interfaces, allowing it to inherit the abstract methods declared in each interface.
	Example:
	`// Define two interfaces with abstract methods`
	`interface Interface1 {`
		[5]	CO3	L2, L3

	void method1();
	}
	
	interface Interface2 {
	void method2();
	}
	
	// Implement both interfaces in a class
	class MyClass implements Interface1, Interface2 {
	@Override
	public void method1() {
	System.out.println("Implementing method1 from Interface1");
	}
	
	@Override
	public void method2() {
	System.out.println("Implementing method2 from Interface2");
	}
	
	// Additional methods specific to MyClass
	public void additionalMethod() {
	System.out.println("Additional method in MyClass");
	}
	}
	
	public class Main {
	public static void main(String[] args) {
	// Create an instance of MyClass
	MyClass myObject = new MyClass();
	
	// Call methods from both interfaces
	myObject.method1();
	myObject.method2();
	
	// Call an additional method from MyClass
	myObject.additionalMethod();
	}
	}			

2 (a)	Does the below Java code with abstract method compile? If yes provide implementation for method. If no write the reason and correct the error. class Puppy { abstract void showName();} **Solution:** **No.** **Correct code:**	[5]	CO3	L3

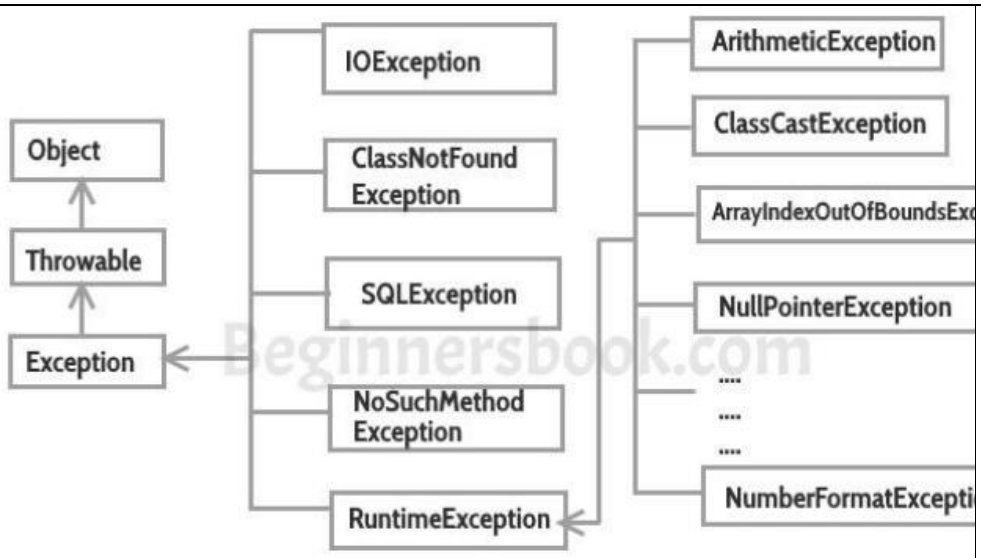
	Class should be abstract Abstract class Puppy{abstract void showName();}			
2(b)	What is overloading. Write the difference between method overloading and constructor overloading. (Min 4 points). **Solution:** In Java, overloading refers to the ability to define multiple methods or constructors in a class with the same name but with different parameters. This allows a class to have multiple methods or constructors that perform similar actions but can handle different types or numbers of arguments. Overloading is based on the concept of polymorphism and is a fundamental feature of object-oriented programming **Method Overloading:** 1. Involves defining multiple methods in a class with the same name but different parameter lists. 2. Used for providing variations of a method based on the type or number of parameters. 3. Return type alone is not sufficient to differentiate overloaded methods. 4. Can be inherited by subclasses. **Constructor Overloading:** 1. Involves having multiple constructors in a class with different parameter lists. 2. Used for creating objects with different initial states. 3. Constructors do not have a return type. 4. Not inherited, but can be called using the super() keyword in subclasses.	[5]	CO3	L2

3(a).	What is the output of the below Java program with an abstract class?
	final abstract class Bell
	{ }
	class DoorBell extends Bell{
	DoorBell()
	{System.out.println("DoorBell ringing..");}}
	public class AbstractClassTesting2
	{public static void main(String[] args){
	Bell bell = new DoorBell();}}
	Output: Compile Time Error
	Reason: Final classes can not be inherited.
		[5]	CO3	L3

3(b)	Write a java program to create one interface CreditCard with 2 methods accptRupees() and acceptDoller(). Provide implementation for both methods and print the output.
	Solution :
	// Define the CreditCard interface
	interface CreditCard {
	void acceptRupees(double amount);
	void acceptDollars(double amount);
	}
		[5]	CO3	L3

	// Implement the CreditCard interface
	class CreditCardImpl implements CreditCard {
	@Override
	public void acceptRupees(double amount) {
	System.out.println("Accepted Rupees: " + amount);
	}
	
	@Override
	public void acceptDollars(double amount) {
	System.out.println("Accepted Dollars: " + amount);
	}
	}
	
	// Main class to demonstrate the program
	public class Main {
	public static void main(String[] args) {
	// Create an instance of CreditCardImpl
	CreditCardImpl myCreditCard = new CreditCardImpl();
	
	// Call acceptRupees() method
	myCreditCard.acceptRupees(5000.75);
	
	// Call acceptDollars() method
	myCreditCard.acceptDollars(100.50);
	}
	}			

4(a)	Explain the hierarchy of exceptions in Java. How are checked and unchecked exceptions related in the hierarchy with diagram? **Solution:** In Java, exceptions are categorized into a hierarchy based on the inheritance structure defined by the Throwable class. The two main types of exceptions in this hierarchy are: **Checked Exceptions (Compile-time Exceptions):** • These exceptions are checked at compile time. • Subclasses of Exception that are not subclasses of RuntimeException. • Developers are required to handle or declare these exceptions using the try-catch block. • Common checked exceptions include IOException, SQLException, and FileNotFoundException. **Unchecked Exceptions (Runtime Exceptions):** • These exceptions are not checked at compile time and typically result from programming errors or unexpected conditions at runtime. • Subclasses of RuntimeException. • Developers are not required to handle or declare these exceptions explicitly. • Common unchecked exceptions include NullPointerException, ArrayIndexOutOfBoundsException, and ArithmeticException. • The Throwable class is the root class for the exception hierarchy.	[5]	CO4	L2

4(b)	What is the output of the below Java code?
	public class ExceptionTest5{
	public static void main(String[] args) {
	int ary[] = new int[2];
	ary[10] = 5;
	try { int number= 2/0;}
	catch(Exception e){ System.out.println("Divide by Zero"); }
	finally{System.out.println("Inside FINALLY block"); }}}
	
	Output: Inside FINALLY block
	Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 10 out of bounds for length 2
	at ExceptionTest5.main(ExceptionTest5.java:5)
		[5]	CO3	L3

5 (a)	How is super used to call the constructor of the super class? Write program to call super class constructor.
	Solution:
	In Java, the super keyword is used to call the constructor of the superclass. This is typically done within the constructor of a subclass to invoke the constructor of its immediate superclass. The super() statement must be the first statement in the constructor of the subclass.
	Example:
	class Animal {
	String type;
	
	// Constructor of the superclass
		[5]	CO3	L3

```
    Animal(String type) {
        this.type = type;
        System.out.println("Animal constructor called");
    }

    void displayInfo() {
        System.out.println("Type of animal: " + type);
    }
}

class Dog extends Animal {
    String breed;

    // Constructor of the subclass
    Dog(String type, String breed) {
        // Calling the constructor of the superclass using super
        super(type);
        this.breed = breed;
        System.out.println("Dog constructor called");
    }

    void displayBreed() {
        System.out.println("Breed of dog: " + breed);
    }
}

public class Main {
    public static void main(String[] args) {
        // Creating an instance of the subclass Dog
        Dog myDog = new Dog("Mammal", "Labrador");

        // Calling methods from both the superclass and subclass
        myDog.displayInfo();
        myDog.displayBreed();
    }
}
```

5 (b)	Write a java program to use static and default methods inside the interface and access them.
	Program :
	interface MyInterface {
	static void staticMethod() {
	System.out.println("Static method in the interface");
	}
	
	default void defaultMethod() {
	System.out.println("Default method in the interface");
	}
	
	void abstractMethod();
	}
	
	// Implement the interface in a class
	class MyClass implements MyInterface {
	
	public void abstractMethod() {
	System.out.println("Implemented abstract method");
	}
	}
	
	public class Main {
	public static void main(String[] args) {
	// Call the static method using the interface name
	MyInterface.staticMethod();
	
	MyClass myObject = new MyClass();
	
	myObject.defaultMethod();
	
	myObject.abstractMethod();
	}
	}
		[5]	CO4	L3

6(a)	**An object of multi-level inherited abstract class cannot be created in memory?** **State True / false** **Solution:** True	[1]	CO3	L2
6(b)	Choose a correct statement about Java Interfaces? A) Interface contains only abstract methods by default. B) A Java class can implement multiple interfaces C) An Interface can extend or inherit another Interface. D) All the above **Solution: D**	[1]	CO3	L1
6(c)	Which is the correct syntax to import a Java package below? A) import PACKAGE1.*; B) import PACKAGE1.CLASS1; C) import PACKAGE1.PACKAGE2.PACKAGE3.*; D) All the above	[1]	CO4	L1

	Solution : D			
6(d)	In java, can an abstract class be instantiated. Yes or no. **Solution: No**	[1]	CO3	L1
6(e)	In abstract class we can write abstract methods and non abstract methods. True or false. **Solution: True**	[1]	CO3	L1
6(f)	Which keyword used to declare a package? **Solution:Package**	[1]	CO4	L1
6(g)	Finally block is related to Exception in java. True or false. **Solution: True**	[1]	CO4	L1
6(h)	Write syntax to call super class methods using super keyword. **Solution: Super.method_name();**	[1]	CO3	L1
6(i)	Write difference between Exception and Errors. **Solution:** Exceptions are events that occur during the execution of a program and can be handled programmatically, while errors are typically unrecoverable and arise from critical failures in the system or the application.	[1]	CO4	L1
6(j)	Method overriding is a compile time polymorphism. True/False **Solution: True**	[1]	CO3	L1

CI **CCI** **HOD**

PO Mapping

CO-PO and CO-PSO Mapping																				
Course Outcomes		Blooms Level	Modules covered	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
CO1	Analyze the performance of the algorithms, state the efficiency using asymptotic notations, and analyze mathematically the complexity of the algorithm.	L2	M1	3	3	2	3	2	-	-	-	-	-	-	2	-	-	-	-
CO2	Apply divide and conquer approaches and decrease and conquer approaches in solving the problems and analyze the same	L3	M2	3	3	2	3	2	-	-	-	-	-	-	2	-	-	-	-
CO3	Apply the appropriate algorithmic design technique like the greedy method, transform and conquer approaches and compare the efficiency of algorithms to solve the given problem.	L3	M3	3	3	2	3	2	-	-	-	-	-	-	2	-	-	-	-
CO4	Apply and analyze dynamic programming approaches to solve some problems. and improve an algorithm's time efficiency by sacrificing space.	L3	M4	3	3	2	3	2	-	-	-	-	-	-	2	-	-	-	-
CO5	Apply and analyze backtracking, branch and bound methods to describe P, NP, and NP-complete problems.	L3	M5	3	2	2	3	2	-	-	-	-	-	-	2	-	-	-	-

COGNITIVE LEVEL	REVISED BLOOMS TAXONOMY KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PROGRAM OUTCOMES (PO), PROGRAM SPECIFIC OUTCOMES (PSO)				CORRELATION LEVELS	
PO1	Engineering knowledge	PO7	Environment and sustainability	0	No Correlation
PO2	Problem analysis	PO8	Ethics	1	Slight/Low
PO3	Design/development of solutions	PO9	Individual and team work	2	Moderate/ Medium
PO4	Conduct investigations of complex problems	PO10	Communication	3	Substantial/ High
PO5	Modern tool usage	PO11	Project management and finance		
PO6	The Engineer and society	PO12	Life-long learning		
PSO1	Develop applications using different stacks of web and programming technologies				
PSO2	Design and develop secure, parallel, distributed, networked, and digital systems				
PSO3	Apply software engineering methods to design, develop, test and manage software systems.				
PSO4	Develop intelligent applications for business and industry				