2021 Scheme

1	N	T							

NITTE EDUCATION TRUST

NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

An Autonomous Institution Approved by UGC/AICTE/Govt. of Karnataka
Accredited by NBA (Tier – I) and NAAC 'A+' Grade
Affiliated to Visveswaraya Technological University, Belagavi
Post Box No. 6429, Yelahanka, Bengaluru – 560 064, Karnataka, India

KNOWLEDGE • CHARACTER • UNITY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MSE 1 SCHEME AND SOLUTION

Course Title with code	OOP with JAVA(21CS35)	**Maximum Marks**	**30 Marks**
Date and Time	17/12/2022, 9.30-10.30am	**No. of Hours**	1.0
Course Instructor(s)	Dr. Vijaya Shetty S/Ms. Shruthi Shetty/Sowmya P		
1. Answer any **two full questions.** 2. Any missing data may assume suitably.			

Q. No	Question	MAX MARKS

1. a	Employee **compare**(Employee E)
	{
	if(age>E.age)
	return this;
	else
	return E;
	}
		2

1. b	**3 OOP Principles explanation with an example for each-2×3=6m** 1. Encapsulation 2. Inheritance 3. Polymorphism	6

1. c	**Student class definition—4m**
	Main() method------3m
	
	package UNIT1;
	import java.util.Scanner;
	class Student1{
	private String name,USN,Branch;
	private int age;
	private double CGPA;
	Student1(String n,String u, String b,**int** a, **double** c)
	{name=n;USN=u;Branch=b;age=a;CGPA=c;}
	void PrintStudent() {
	System.***out***.println("Name:"+ name);
	System.***out***.println("USN:"+ USN);
	System.***out***.println("Branch:"+ Branch);
	System.***out***.println("Age:"+ age);
	System.***out***.println("CGPA:"+ CGPA);
		7

```
        }
        void CseNmit()
        { if (CGPA>=9 && Branch.equals("CSE"))
                {PrintStudent();
          System.out.println("-------------------------------------------------");}
        }

}
public class StudentArry {
public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int N;
        System.out.println("Enter N value");
        N = sc.nextInt();
        Student1 S[]=new Student1[N];
       for(int i=0; i<N;i++){
          sc.nextLine();
          System.out.println("enter Name,USN,Branch age and CGPA");
          S[i]=new Student1(sc.nextLine(),  sc.nextLine(),sc.nextLine(), sc.nextInt(),
          sc.nextDouble());
        }
       System.out.println("Details of  CSE Students with CGPA>=9");
       for(int i=0; i<N;i++)
        {
                S[i].CseNmit();
        }
 }
}
```

2. a

final for variable 1m, method and class 2*2, 1+2+2=5

The keyword **final** has three uses.

To Declare a named constant

A field can be declared as **final**. Doing so prevents its contents from being modified, making it, essentially, a constant. We can do this in one of two ways:

- First, we can give it a value when it is declared.
- Second, we can assign it a value within a constructor.

Eg: final int FILE_NEW = 1;
final int FILE_OPEN = 2;

Using final to Prevent Overriding

To disallow a method from being overridden, specify **final** as a modifier at the start of its declaration. Methods declared as **final** cannot be overridden. The following fragment illustrates **final**:

```
class A {
final void meth () {
System.out.println("This is a final method.");
}
}
class B extends A {
void meth() { // ERROR! Can't override.
System.out.println("Illegal!");
}
```

5

	}
	Using final to Prevent Inheritance
	Sometimes it is required to prevent a class from being inherited. To do this, precede the class declaration with **final**. Declaring a class as **final** implicitly declares all its methods as **final.**
	Here is an example of a **final** class:
	final class A {
	//...
	}
	// The following class is illegal.
	class B extends A {// ERROR! Can't subclass A
	//...
	}
	As the comments imply, it is illegal for **B** to inherit **A** since **A** is declared as **final**.	

2. b	**Class definition(1m) + inheritance(2m)**
	Main() method------2m
	
	abstract class F1{
	int X, Y;
	F1(int a,int b){
	X=a;
	Y=b;
	}
	abstract void compute();
	}
	class R1 extends F1{
	R1(int a,int b){
	super(a,b);
	}
	void compute(){
	int sum=X+Y;
	System.out.println("Sum is:" + sum);
	}}
	class T1 extends F1{
	T1(int a,int b){
	super(a,b);
	}
	void compute(){
	int result=(X*Y)/2;
	System.out.println("Result is:" + result);
	}}
	class AbstactDemo{
	public static void main(String args[]){
	R1 r=new R1(5,5);
	T1 t=new T1(7,3);
	F1 f;
	f=r;
	f.compute();
	f=t;
	f.compute();
	}}
		5

2. c	**Class definition(1m)+inheritance(2m)**	5

	Main() method------2m
	class Animal
	{
	void noun(){
	System.out.println(“Animal is a noun”);
	}
	class Dog extends Animal{
	void pet(){
	System.out.println(“Dog is man’s best friend”);
	}
	class BabyDog extends Dog{
	void Puppy(){
	System.out.println(“Baby Dog is called Puppy”);
	}
	class MultiInheritance{
	public static void main(String args[]);{
	Animal a=new Animal();
	a.noun();
	Dog d=new Dog();
	d.noun();
	d.pet();
	BabyDog B=new BabyDog();
	B.noun();
	B.pet();
	B.Puppy();
	}}	

3. a	**Garbage Collection - 1.5 M**
	finalize() - 1.5 M
	
	In Java, objects are deallocated automatically using the technique called Garbage Collection. When no references to an object exist, that object is assumed to be no longer needed and the memory occupied by the object is reclaimed by Garbage Collection. Garbage collection only occurs occasionally during the execution of a program and will not occur simply because one or more objects exist that are no longer used.
	The Java runtime calls the finalize() method whenever it is about to recycle an object of a class. Inside the finalize( ) method, all actions that must be performed before an object is destroyed will be specified.
	
	The finalize( ) method has this general form
	protected void finalize()
	{
	//finalization code here
	}
	finalize( ) is called just prior to garbage collection.
		3

3. b	**class Book - 4M** **main method - 2M**	6

	class Book
	{
	String title, author, publisher;
	double cost;
	void input()
	{
	Scanner sc=new Scanner(System.in);
	System.out.println(“Enter title”);
	title= sc.nextLine();
	System.out.println(“Enter author”);
	author= sc.nextLine();
	System.out.println(“Enter publisher”);
	publisher= sc.nextLine();
	System.out.println(“Enter cost”);
	cost= sc.nextDouble();
	}
	
	void display()
	{
	System.out.println(“Title is:” +title);
	System.out.println(“Author is:” +author);
	System.out.println(“Publisher is:” +publisher);
	System.out.println(“Cost of book is:” +cost);
	}
	
	public static void main(String args[])
	
	{
	Book b= new Book();
	b.input();
	b.display();
	}
	}	

3. c	**Bank interface - 1M**
	SBI class - 2M
	PNB class - 2M
	main method - 1M
	
	interface Bank
	{
	float rateofinterest();
	}
	
	class SBI implements Bank
	{
	float rateofinterest()
		6

```
 {
   Return  8.7f;
  }

class PNB implements Bank
{
 float rateofinterest()
 {
  Return 6.8f;
}

class Tester
{
 public static void main(String args[])
{
  Bank b;
  b = new SBI();
  System.out.println("Rate of interest in SBI :" + b.rateofinterest());
  b= new PNB();
  System.out.println("Rate of interest in PNB i:" + b.rateofinterest();
}
}
```

Faculty Signature	Course Co-Ordinator/Mentor Signature	HoD Signature