Exercise-9

**Strings**

**Q1. Define String**

**Ans:** Strings, which are widely used in Java programming, are asequence of characters. In the Java programming language, strings are objects. The Java platform provides the String class to create and manipulate strings.

**Q2. List 10 methods of string**

**Ans**: 1. charAt(); 2.toUpperCase(); 3.toLowerCase(); 4.compareTo(); 5.concat(); 6.endsWith(); 7.equals(); 8.indexOf(); 9.isEmpty(); 10.substring();

**Q3. Write three different ways of creating string**

**Ans:**

* Using String literal. You can create String objects with String literal. String str="Hello!";
* Using new keyword. This is the common way to create a String object in java. ...
* Using character array. You could also convert character array into String here.

**Q4. Write the difference between String and String Buffer object**

**Ans:** In Java programming language, strings are treated as objects. The Java platform provides the String class to create and manipulate strings. Whereas, String Buffer class is a thread-safe**,** mutable sequence of characters. A string buffer is like a String, but can be modified.

**Q5. What it is require using equals method of the string object to compare to strings**.

**Ans:** The equals() method compares two strings, and returns true if the strings are equal, andfalse ifnot. Tip: Use the compareTo() method to compare two strings lexicographically.

**Q6. What is the output of the below code and give the reason why it is so.**

**String s1="cme";**

**String s2="cme";**

**System.out.println(s1==s2);**

**Ans**: The output is true.

**Q7. What is the output of the below code and give the reason why it is so.**

**String s1=new String("cme");**

**String s2=new String("cme");**

**System.out.println(s1==s2);**

**Ans**: The output is false .

**Q8. What is the output of the below code and give the reason why it is so.**

**String s1="cme";**

**String s2="cme";**

**System.out.println(s1.equals(s2));**

**Ans**: The output is true.

**Q9. What is the output of the below code and give the reason why it is so.**

**System.out.println(“java is wonderful ”+10+20);**

**Ans :** The output is java iswonderful 1020.

**Q10. What is the output of the below code and give the reason why it is so.**

**System.out.println(10+20+ “java is wonderful”);**

**Ans :** The output is 30 java iswonderful .

**PROGRAMS**

**Q1. Write a Java program to compare two strings lexicographically. Two strings are lexicographically equal if they have the same length and contain the same characters same positions. Ignore case differences. (Algorithm)**

**Algorithm:**

* Start.
* Declare two strings.
* Initialize the strings.
* First, compare the strings using the equals() method.
* Print the result.
* Now, again compare by using the .equals() operator.
* Here, compare the string and the value stored in another string.
* Display the output.

**Program:**

import java.util.Scanner;

class IgnoringCase

{

public static void main(String s[])

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter the String=");

String s1=sc.nextLine();

System.out.println("Enter the Another String=");

String s2=sc.nextLine();

if(s1.length()==s2.length())

{

System.out.println("Lengths are equal \nChecking letters\n");

if(s1.equalsIgnoreCase(s2))

System.out.println("The Two Strings are equal");

else

System.out.println("The Two Strings are not equal");

}

else

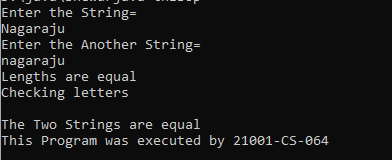
System.out.println("The strings are not equal");

System.out.println("This Program was executed by 21001-CS-064");

}

}

**OUTPUT:**

****

**Q2. Write a Java program to compare two strings lexicographically. Two strings are lexicographically equal if they have the same length and contain the same characters in the same positions. Do not ignore case differences.**

**PROGRAM:**

import java.util.Scanner;

class CaseSensitive

{

public static void main(String s[])

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter the String=");

String s1=sc.nextLine();

System.out.println("Enter the Another String=");

String s2=sc.nextLine();

if(s1.length()==s2.length())

{

System.out.println("Lengths are equal \nChecking letters\n");

if(s1.equals(s2))

System.out.println("The Two Strings are equal");

else

System.out.println("The Two Strings are not equal");

}

else

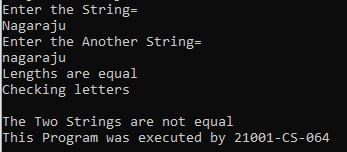
System.out.println("The strings are not equal");

System.out.println("This Program was executed by 21001-CS-064");

}

}

**OUTPUT:**

****

**Q3. Write a Java program to test if a given string contains the specified sequence of char values.. (Algorithm)**

**ALGORITHM:**

* Enter the String and char array to search
* Run the for loops for both string and char array
* Check each character in the char array with the string elements
* If matches, print the letter
* We can check the each character of the string by the string method called charAt();

**PROGRAM:**

import java.util.Scanner;

class StringCheck

{

public static void main(String s[])

{

int c=0,k;

Scanner sc=new Scanner(System.in);

char[] ch=new char[]{'i','a','g','r','j'};

String s1=new String(ch);

System.out.println("The characters in the array are i a g r j ");

System.out.println("Enter the String=");

String s2=sc.nextLine();

out:for(int i=0;i<s1.length();i++)

{

c=0;

for(int j=0;j<s2.length();j++)

{

if(s2.charAt(j)==s1.charAt(i))

{

c++;

System.out.println("The letters which are present are="+s1.charAt(i));

continue out;

}

}

if(c==0)

{

System.out.println("The letters which are not present="+s1.charAt(i));

}

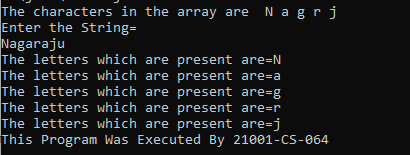
}

System.out.println("This Program Was Executed By 21001-CS-064");

}

}

**OUTPUT:**

****

**Q4. Write a Java program to check whether a given string ends with the contents of another string.**

PROGRAM:

import java.util.Scanner;

class EndsWith

{

public static void main(String s[])

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter the String=");

String s1=sc.nextLine();

System.out.println("Enter the another to check the String=");

String s2=sc.nextLine();

if(s1.endsWith(s2))

System.out.println("The String end matches");

else

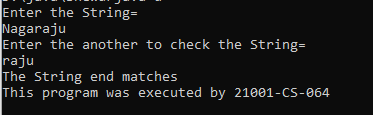
System.out.println("The string end doesnt matches");

System.out.println("This program was executed by 21001-CS-064");

}

}

**OUTPUT:**

****

**Q5. Write a Java program to check whether a given string starts with the contents of another string.**

**PROGRAM:**

import java.util.Scanner;

class StartsWith

{

public static void main(String s[])

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter the String=");

String s1=sc.nextLine();

System.out.println("Enter the another to check the String=");

String s2=sc.nextLine();

if(s1.startsWith(s2))

System.out.println("The String starting matches");

else

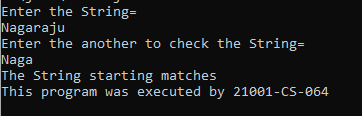
System.out.println("The string starting doesnt matches");

System.out.println("This program was executed by 21001-CS-064");

}

}

**OUTPUT:**

****

**Q6. Write a Java Program to prove that string is immutable.**

**PROGRAM:**

import java.util.Scanner;

class Immutable

{

public static void main(String s[])

{

String s1;

Scanner sc=new Scanner(System.in);

System.out.println("Enter the string=");

s1=sc.nextLine();

System.out.println("The entered string is="+s1);

System.out.println("Trying to concat with programming");

s1.concat("programming");

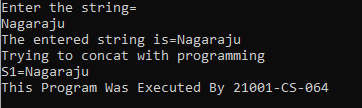
System.out.println("S1="+s1);

System.out.println("This Program Was Executed By 21001-CS-064");

}

}

**OUTPUT:**

****

**Q7. Write a Java Program to prove that StringBuffer is mutable**

**PROGRAM:**

class StringBufferExample

{

public static void main(String s[])

{

StringBuffer sb=new StringBuffer("Hello ");

sb.append("Java");

System.out.println(sb);

System.out.println("This Program Was Executed By 21001-CS-064");

}

}

**OUTPUT:**

****

**Q8. Write a Java Program to check a string is a palindrome or not without using library function, without reversing it and another string object. (Algorithm)**

**ALGORITHM:**

* Given string to convert char array.
* Take two point. 1st beginning of the string a[0] 2nd one ending of the string a[length-1]
* Compare every char in char array. Compare beginning to end and ending begin while looping.

**PROGRAM:**

import java.util.Scanner;

class StringPalindrome

{

public static void main(String s3[])

{

int c=0;

Scanner sc=new Scanner(System.in);

System.out.println("Enter the string=");

String s=sc.nextLine();

out:for(int i=0;i<=(s.length())/2;i++)

{

for(int j=s.length()-1;j>=(s.length())/2;j--)

{

if(s.charAt(i)==s.charAt(j))

c++;

}

}

if(c>=s.length()/2)

System.out.println("Palindrome");

else

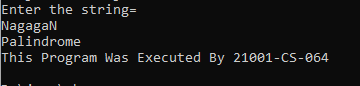
System.out.println("Not palindrome");

System.out.println("This Program Was Executed By 21001-CS-064");

}

}

**OUTPUT:**

****

**Exercise-10**

**Inheritance**

**Q1. Define Inheritance?**

Ans. Inheritance in Java is a concept that acquires the properties from one class to other classes.

for example: the relationship between father and son. Inheritance in Java is a process of acquiring all the behaviours of a parent object**.**

**Q2. List the types of inheritance?**

Ans.

· Single Inheritance.

· Multiple Inheritance.

· Multilevel Inheritance.

· Hierarchical Inheritance.

· Hybrid Inheritance.

**Q3. Define Single Level Inheritance?**

Ans. The inheritance in which a single derived class is inherited from a single base class is known as the Single Inheritance. It is the simplest among all the types of inheritance since it does not include any kind of inheritance combination or different levels of inheritance.

**Q4. Define Multi Level Inheritance?**

Ans. Multilevel Inheritance in java occurs when a class extends a class that extends another class. This is called multilevel Inheritance in java. For example, class C extends class B, and class B extends class A

**Q5. Define Hierarchical Inheritance?**

Ans. In Hierarchical inheritance, more than one sub-class inherits the property of a single base class. There is one base class and multiple derived classes. Several other classes inherit the derived classes as well. Hierarchical structures thus form a tree-like structure.

**Q6 .Define Interface?**

Ans. An interface in Java is a blueprint of a class. It has static constants and abstract methods. The interface in Java is a mechanismto achieve abstraction

**Q7. Give reason why java does not support multiple inheritance through classes.?**

Ans. Java doesn't support multiple inheritances in classes because it can lead to diamond problem and rather than providing some complex way to solve it, there are better ways through which we can achieve the same result as multiple inheritances

**Q8 Give reason why java supports multiple inheritance through interfaces..?**

Ans. multiple inheritance is not supported in the case of class because of ambiguity. However, it is supported in case of an interface because there is no ambiguity. It is because its implementation is provided by the implementation class.

**Q9. What are the contents of interface in java version 6?**

Ans.In java 6,an interface is a reference type that specifies a set of methods that a class implementing the interface must implement.It can also include and defult methods.

**Q10.what are thw contents of interface in java version 8?**

Ans.In java 8,an interface is similar to that in java 6 and still specifies a set of methods that a class implementing tha interface must implement .However ,java 8 introduced several new features to interface that were not present in java6

· Default methods

· Static methods

· Private methods

· Do something

· Getvalue

**Q11. What are the contents of interface in java version 9?**

Ans. In java 9, an interface is similar to that in Java 8 and still specifies a set of methods that a class implementing the interface must implement. However, Java 9 introduced several new features to interfaces that were not present in Java .

· Private methods

· Private static methods

**Q12. What is the keyword used to implement an interface?**

Ans. The implements keyword is used to implement an interface . The interface keyword is used to declare a special type of class that only contains abstract methods. To access the interface methods, the interface must be "implemented" (kinda like inherited) by another class with the implements keyword (instead of extends ).

**Q13. What is the keyword used to extends an interface.?**

Ans. Extends: In Java, the extends keyword is used to indicate that the class which is being defined is derived from the base class using inheritance. So basically, extends keyword is used to extend the functionality of the parent class to the subclass.

**Q15. Why it is required for the implementing class implementing two interfaces having**

**same method to override the method that have default implementation in their corresponding**

**interface.?**

Ans.It is required for the implementing class to override the default menthods when it implements two intrfaces that have the same methods with a default implementation because the class must provide a concreate implementation for the methods in order to satisfy both interfaces.

**Q16. Why it is not required for the implementing class implementing two interfaces having**

**same method to override the method that have default implementation in their corresponding**

**interface.?**

Ans.It is required for the implementing class to override the default methods when it implements two interfaces that have tha same methods with a default implementation. This is because the class must provide a concreate implementation for the method in order to satisfy both interfaces.

**Q17. Why Java 9 interfaces have implemented private methods?**

Ans. An interface can have private methods since Java 9 version. These methods are visible only inside the class/interface, so it's recommended to use private methods for confidential code. That's the reason behind the addition of private methods in interfaces.

**Q18. Nested interfaces are static are non-static.?**

Ans.Nested interfaces are static.

**PROGRAMS**

**Q1. Define a Person with attributes name, age and gender and a method display\_details function. Then define three different classes namely Pilot, Teacher and a Doctor that inherits Person class, add to these classes respective attributes and methods. In the table given below attributes of different profession are mentioned, you add at least two services given by each of these individual. Name the type of inheritance that should be employed in this case.**

|  |  |  |
| --- | --- | --- |
| **PILOT** | **TEACHER** | **DOCTOR** |
| Airline | Subject | Hospital |
| Hours(no of Hours in air) | College | Experience |
| Home Airport | No.of Students | Specialization |
| Destination Airport | Workload | Daily Patients |
| Flight Size | Semester | Timings |

**SERVICES**

|  |  |  |
| --- | --- | --- |
|  | Teaches |  |
|  | mentor |  |

**PROGRAM:**

import java.util.Scanner;

class Person

{

String Name;

String gender;

int age;

Person(int a,String n,String g )

{

Name = n;

age=a;

gender=g;

}

public void display()

{

System.out.println("name : "+Name);

System.out.println("gender : " +gender);

System.out.println("age : "+age);

}

}

class Doctor extends Person

{

String hos;

int stimings;

int etimings;

int ex;

String spc;

int dp;

public Doctor(int a,String n,String g,String h,int st,int et,int e,String s,int op)

{

super(a,n,g);

hos=h;

stimings=st;

etimings=et;

ex=e;

spc=s;

dp=op;

}

public void display()

{

super.display();

System.out.println("hospital : "+hos);

System.out.println("timing : "+stimings+" AM to "+etimings+" PM");

System.out.println("daily patients:"+dp);

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

}

}

class Teacher extends Person

{

String sub;

String clg;

int stud;

int sem;

public Teacher(int a,String n,String g,String su,String cl,int stu,int se)

{

super(a,n,g);

sub=su;

clg=cl;

stud=stu;

sem=se;

}

public void display()

{

super.display();

System.out.println("subject : "+sub);

System.out.println("college : "+clg);

System.out.println("number of students : "+stud);

System.out.println("semester : "+sem);

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

}

}

class Pilot extends Person

{

String airl;

int hrs;

String hairp;

String dairp;

int sizef;

public Pilot(int a,String n,String g,String ai,int hr,String hair,String dair,int szf)

{

super(a,n,g);

airl=ai;

hrs=hr;

hairp=hair;

dairp=dair;

sizef=szf;

}

public void display()

{

super.display();

System.out.println("airlines : "+airl);

System.out.println("hours on flight : "+hrs);

System.out.println("home airport : "+hairp);

System.out.println("destination airport : "+dairp);

System.out.println("size of plane : "+sizef);

}

}

class MainMethod

{

public static void main(String s[])

{

Doctor d=new Doctor(28,"Nagaraju","male","shubham nursing home",10,13,30,"children's",130);

d.display();

Teacher t=new Teacher(30,"sujith","male","english","bharatiya vidya bhavan's",130,2);

t.display();

Pilot p=new Pilot(21,"abhi","male","Air India",16,"Rajiv Gandhi Airport","Bangalore Airport",787);

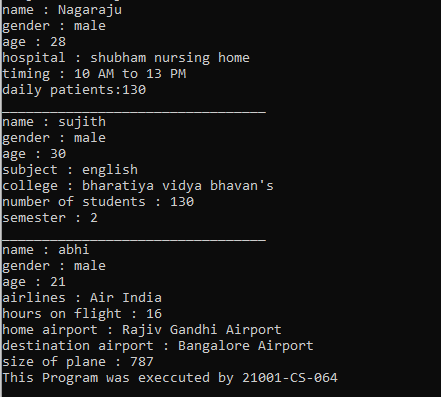
p.display();

System.out.println("This Program was execcuted by 21001-CS-064");

}

}

**OUTPUT:**

****

**Q2. Define a class that implements an interface A. Interface A is extended from B and Interface B is extended from interface C. Each interface has two methods. Mention the type of inheritance.**

**PROGRAM:**

interface c

{

public void first();

public void second();

}

interface b extends c

{

public void first1();

public void second1();

}

interface a extends b

{

public void first2();

public void second2();

}

class Interface implements a

{

public void first()

{

System.out.println("This is first method of first interface");

}

public void second()

{

System.out.println("This is second method of first interface"); }

public void first1()

{

System.out.println("This is first method of second interface");

}

public void second1()

{

System.out.println("This is second method of second interface");

}

public void first2()

{

System.out.println("This is first method of third interface");

}

public void second2()

{

System.out.println("This is second method of third interface");

}

}

class MainMethod

{

public static void main(String s[])

{

Interface Inref=new Interface();

Inref.first();

Inref.second();

Inref.first1();

Inref.second1();

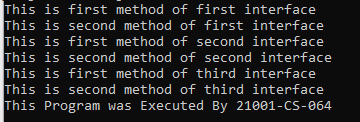
Inref.first2();

Inref.second2();

System.out.println("This Program was Executed By 21001-CS-064");}

}

**OUTPUT:**

****

**Q3. Define a class that implements an interface A. Interface A is extended from B and C interfaces. Each interface has two methods. Mention the type of inheritance.**

**PROGRAM:**

interface b

{

public void first();

public void second();

}

interface c

{

public void first1();

public void second1();

}

interface a extends b,c

{

public void first2();

public void second2();

}

class Interface implements a

{

public void first()

{

System.out.println("This is first method of first interface");

}

public void second()

{

System.out.println("This is second method of first interface");

}

public void first1()

{

System.out.println("This is first method of second interface");

}

public void second1()

{

System.out.println("This is second method of second interface");

}

public void first2()

{

System.out.println("This is first method of third interface");

}

public void second2()

{

System.out.println("This is second method of third interface");

}

}

class MainMethod

{

public static void main(String s[])

{

Interface Inref=new Interface();

Inref.first();

Inref.second();

Inref.first1();

Inref.second1();

Inref.first2();

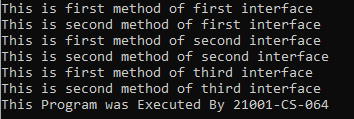
Inref.second2();

System.out.println("This Program was Executed By 21001-CS-064");

}

}

**OUTPUT:**

****

**Q4. Define a class ABC that implements interfaces A, B and C. Each interface has two methods. Mention the type of inheritance.**

**PROGRAM:**interface a

{

public void first();

public void second();

}

interface b

{

public void first1();

public void second1();

}

interface c

{

public void first2();

public void second2();

}

class Interface implements a,b,c

{

public void first()

{

System.out.println("This is first method of first interface");

}

public void second()

{

System.out.println("This is second method of first interface");

}

public void first1()

{

System.out.println("This is first method of second interface");

}

public void second1()

{

System.out.println("This is second method of second interface");

}

public void first2()

{

System.out.println("This is first method of third interface");

}

public void second2()

{

System.out.println("This is second method of third interface");

}

}

class MainMethod

{

public static void main(String s[])

{

Interface Inref=new Interface();

Inref.first();

Inref.second();

Inref.first1();

Inref.second1();

Inref.first2();

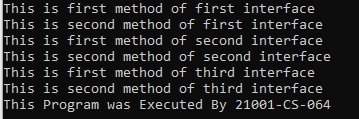
Inref.second2();

System.out.println("This Program was Executed By 21001-CS-064");

}

}

**OUTPUT:**

****

**Q5. Define a class Named Outer implementing IOuter interface and another class Named Inner implementing Inner interface. Inner interface is defined inside Outer interface. Each interface have one method. Call these Methods from the main method of the class named MainClass.**

**PROGRAM:**

interface IOuter

{

void first();

void second();

interface IInner

{

void first1();

void second1();

}

}

class Outer implements IOuter

{

public void first()

{

System.out.println("This is outerinterface first method");

}

public void second()

{

System.out.println("This is outer interface second method");

}

}

class Inner implements IOuter.IInner

{

public void first1()

{

System.out.println("This is inner interface first method");

}

public void second1()

{

System.out.println("This is inner interface second method");

}

}

class MainMethod

{

public static void main(String s[])

{

Inner in=new Inner();

Outer ou=new Outer();

in.first1();

in.second1();

ou.first();

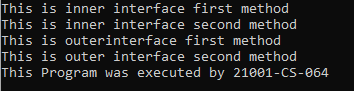
ou.second();

System.out.println("This Program was executed by 21001-CS-064");

}

}

**OUTPUT:**

****

**EXERCISE 11**

**Chaining**

**Q1.Define Constructor.**

**Ans:**A Constructor in java is a special method that is used to intiliase objects.The constructor is called when an object of a class is created.

**2.Define Constructor Overloading.**

**Ans:**Constructor overloading in Java refers to the use of more than one constructor in an instance class. However, each overloaded constructor must have different signatures. For the compilation to be successful, each constructor must contain a different list of arguments.

**3.How many Constructors are there in a class?**

There are two types of constructors parameterized constructors and no-arg constructors.

**4.When default constructor does not exist.**

**Ans:** Java compiler automatically creates a default constructor (Constructor with no arguments) in case no constructor is present in the java class.

**5.Write three uses of this keyword.**

**Ans:**

* this can be used to refer current class instance variable.
* this can be used to invoke current class method (implicitly)
* this() can be used to invoke current class constructor.
* this can be passed as an argument in the method call.
* this can be passed as argument in the constructor call.

**6.Write three uses of super keyword.**

**Ans:**

* It is used to refer to an instance variable of the immediate parent class.
* It is used to invoke a method of the immediate parent class.
* It is used to invoke a constructor of immediate parent class.

**7.Can super and this keywords exits together in the constructor.Give reason**

**Ans:**No, we cannot use the this() and super() in the same constructor. If we want to use this() or super() inside a constructor, they must be written (executed) at first. Both this() and super() cannot be executed simultaneously, hence “this()” and “super()” in Java cannot be used in the same constructor.

**Programs**

**Q1.Write a java program to chain the constructor using this keyword.**

**Program:**

class This

{

This()

{

this(5);

System.out.println("default constructor ");

}

This(int a)

{

this(5.12f);

System.out.println("the value of int : "+a);

}

This(float a)

{

System.out.println("the value of float : "+a);

}

}

class MainMethod

{

public static void main(String s[])

{

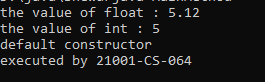
This th=new This();

System.out.println("executed by 21001-CS-064");

}

}

**Output**:



**Q2.Write a java program to chain the constructor using super keyword.**

**Program**

class Super

{

int value;

Super(int a)

{

value=a;

}

public void display()

{

System.out.println("value : "+value);

}

}

class sub extends Super

{

int value2;

sub(int a,int b)

{

super(a);

value2=b;

}

public void show()

{

super.display();

System.out.println("value2 : "+value2);

}

}

class MainMethod

{

public static void main(String s[])

{

sub su=new sub(12,34);

su.show();

System.out.println("Executed by 21001-CS-064");

}

}

**OUTPUT:**

****

**Q3. Write a Java program to uncover the instance variable when the local variable has the same name as the instance variable.**

**Program:**class Variable

{

int length; int breadth;

Variable(int length ,int breadth)

{

this.length=length; this.breadth=breadth;

}

public void display()

{

System.out.println("The length is="+length); System.out.println("The breadth is="+breadth);

}

public static void main(String s[])

{

Variable v=new Variable(5,7); v.display();

System.out.println("This program was executed by 21001-CS-064");

}

}

**OUTPUT:**

****

**Q4. Write a Java program to uncover the instance variable of a super class when the sub class instance variable have the same name as the instance variable of the super class.**

**Program:**

class VariableThis

{

int a=10;

}

class Variable extends VariableThis

{

int a=20; void display()

{

System.out.println("The super class a is="+super.a);

}

public static void main(String s[])

{

Variable v=new Variable(); v.display();

System.out.println("This program was executed by 21001-CS-064");

}

}

**OUTPUT:**

****

**EXERCISE 12**

**POLYMORPHISM**

**Q1.Define Polymorphism**

**Ans**: Polymorphism in Java is a concept by which we can perform a single action in different ways. Polymorphism is derived from 2 Greek words: poly and morphs. The word "poly" means many and "morphs" means forms. So polymorphism means many forms.

**Q2.List the types of Polymorphism**

**Ans:** There are two types of polymorphism in Java: compile time polymorphism and run time polymorphism in java. This java polymorphism is also referred to as static polymorphisms and dynamic polymorphisms.

**Q3.Define static Polymorphism**

**Ans.** Compile-Time Polymorphism Also called static polymorphism, this type of polymorphism is achieved by creating multiple methods with the same name in the same class, but with each one having different numbers of parameters or parameters of different data types.

**Q4.Define Dynamic Polymorphism**

**Ans**: Dynamic polymorphism is a process or mechanism in which a call to an overridden method is to resolve at runtime rather than compile-time. It is also known as runtime polymorphism or dynamic method dispatch. We can achieve dynamic polymorphism by using the method overriding.

**Programs**

**Q1. Write a program of static polymorphism.**

**PROGRAM:**

class Bank

{

float Interest()

{

return 0;

}

}

class SBI extends Bank

{

float Interest()

{

return 9.3f;

}

}

class CANARA extends Bank

{

float Interest()

{

return 7.3f;

}

}

class ICICI extends Bank

{

float Interest()

{

return 9.1f;

}

}

class MainMethod

{

public static void main(String args[])

{

Bank b;

b=new SBI();

System.out.println("SBI Rate of Interest: "+b.Interest());

b=new CANARA();

System.out.println("CANARA Rate of Interest: "+b.Interest());

b=new ICICI();

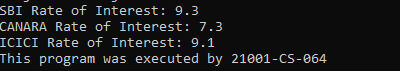
System.out.println("ICICI Rate of Interest: "+b.Interest());

System.out.println("This program was executed by 21001-CS-064");

}

}

**OUTPUT:**



**Q2. Write a program to override a method PROGRAM:**

class Super

{

void display()

{

System.out.println("Iam from the Super class");

}

}

class Sub extends Super

{

void display()

{

System.out.println("Iam from the Sub class");

}

}

class MainMethod

{

public static void main(String s1[])

{

Sub s=new Sub(); s.display();

System.out.println("This Program was executed by 21001-CS-064");

}

}

**OUTPUT:**

****

**Q3.Write a java program of dynamic polymorphism.**

**PROGRAM:**class Base

{

public Base()

{

System.out.println("This is default constructor");

}

public Base(int a,int b)

{

System.out.println("The a value is="+a); System.out.println("The b value is="+b);

}

public Base(float c,float d,float e)

{

System.out.println("The c is="+c);

System.out.println("The d is="+d);

System.out.println("The e is="+e);

}

public static void main(String s[])

{

Base bref=new Base(); Base cref=new Base(1,2);

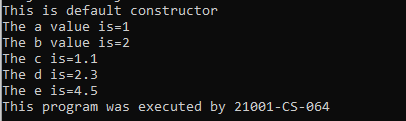
Base dref=new Base(1.1f,2.3f,4.5f);

System.out.println("This program was executed by 21001-CS-064");

}

}

**OUTPUT:**

****

**EXERCISE 13**

**Static And Final**

**Q1.What is the significance of static keyword?**

**Ans:** The static keyword is a non-access modifier used for methods and attributes. Static methods/attributes can be accessed without creating an object of a class.

**Q2.Write the syntax to access static member of a class**

**Ans**: Static member functions are called using the class name.

**Syntax-** class\_name::function\_name( )

**Q3.List three uses of final keyword.**

**Ans:** Once any data member (a variable, method, or class) gets declared as final, it can only be assigned once.

* The final variable cannot be reinitialized with another value.
* A final method cannot be overridden by another method.
* A final class cannot be extended or inherited by another child class.

**Q4.Why it is not possible to access non static members in the static functions.**

**Ans:** Since, static function does not know about object, so, it is impossible for a static function to know on which class object or class instance it is being called. Hence, whenever, we try to call non-static variable from a static function, the compiler throughs an error.

**Q5.Why it is not possible to acess static members in the non static function.**

**Ans:** In the non-static method, the method can access static data members and static methods as well as non-static members and method of another class or same class, also can change the values of any static data member

**Programs**

**Q1.Write a java program on avoid method over riding.**

**Program:**class FinalOver

{

final void display()

{

System.out.println("This is first method");

}

}

class StartOver extends FinalOver

{

void display()

{

System.out.println("This is second method");

}

}

class MainMethod

{

public static void main(String s[])

{

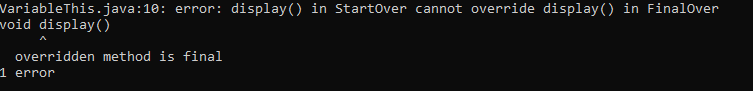
FinalOver fi=new FinalOver(); fi.display();

System.out.println("This program was executed by 21001-CS-064");

}

}

**Output:**



**Q2.Write a java program on avoid method inheritance.**

**Program:**final class Super

{

void display()

{

System.out.println("Iam from super class");

}

}

class Sub extends Super

{

void display()

{

System.out.println("Iam from Sub class");

}

}

class MainMethod

{

public static void main(String s[])

{

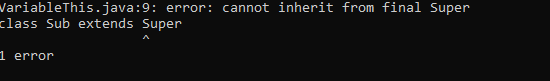
Sub su=new Sub(); su.display();

System.out.println("This program was executed by 21001-CS-064");

}

}

**Output:**



**Q3. Write a Java program to find the area of the circle where PI is declared as constant.**

**PROGRAM:**import java.util.Scanner;

class Area

{

public static void main(String s[])

{

final float PI=3.14f;

Scanner sc=new Scanner(System.in);

System.out.println("Enter the radius=");

int r=sc.nextInt();

System.out.println("The PI value is="+PI);

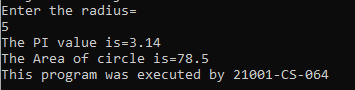
System.out.println("The Area of circle is="+(PI\*(r\*r)));

System.out.println("This program was executed by 21001-CS-064");

}

}

**Output:**

****