Ex:26 **IS-A RELATIONSHIP**

**package** javaset2;

**class**regi{

**float**salary=40000;

}

**class** javalab026 **extends**regi {

**int**bonus=10000;

**publicstaticvoid**main(String args[]){

javalab026 p=**new** javalab026();

System.***out***.println("Programmer salary is:"+p.salary);

System.***out***.println("Bonus of Programmer is:"+p.bonus);

}

}

Output:

Programmer salary is:40000.0

Bonus of Programmer is:10,000.0

Ex:27 **JAVA UNSUPPORTS MULTIPLE INHERITANCE**

**package** javaset2;

**publicclass** javalab027 {

**void**msg(){System.***out***.println("Hello");}

}

**class**java{

**void**msg(){System.***out***.println("Welcome");}

}

**class** C **extends** javalab027,java{//suppose if it were

**public**Static**void**main(String args[]){

C obj=**new**C();

obj.msg();//Now which msg() method would be invoked?

}

}

Output:

Error: due to inablility to do multiple inheritance

Ex:28 **OVERLOAD MAIN METHOD**

**package** javaset2;

**publicclass** javalab028 {

**int**square(**int**n){

**return**n\*n;

}

}

**class**Circle{

javalab028 op;//aggregation

**double**pi=3.14;

**double**area(**int**radius){

op=**new** javalab028();

**int**rsquare=op.square(radius);//code reusability (i.e. delegates the method call).

**return**pi\*rsquare;

}

**publicstaticvoid**main(String args[]){

Circle c=**new**Circle();

**double**result=c.area(5);

System.***out***.println(result);

}

}

Output:

78.5

Ex:29 **PROBLEM WITHOUT OPERATOR OVERRIDING**

**package** javaset2;

**publicclass** javalab029 {

**void**run(){System.***out***.println("Vehicle is running");}

}

**class** Bike **extends**javalab029{

**publicstaticvoid**main(String args[]){

Bike obj = **new**Bike();

obj.run();

}

}

Output:

Vehicle is running

Ex:30 **METHOD OVERRIDING**

**package** javaset2;

**class** javalab030{

**void**run(){System.***out***.println("Vehicle is running");}

}

**class** Bike2 **extends** javalab030{

**void**run(){System.***out***.println("Bike is running safely");}

**publicstaticvoid**main(String args[]){

Bike2 obj = **new** Bike2();

obj.run();

}

}

Output:

Bike is running safely

Ex:31 **PROBLEM WIHTOUT SUPER KEYWORD**

**package** javaset2;

**publicclass** javalab031 {

**int**speed=50;

}

**class** Bike3 **extends**javalab031{

**int**speed=100;

**void**display(){

System.***out***.println(speed);//will print speed of Bike

}

**publicstaticvoid**main(String args[]){

Bike3 b=**new** Bike3();

b.display();

}

}

Output:

50

Ex:32 **PROBLEM WITH SUPER KEYWORD**

**package** javaset2;

**publicclass** javalab032 {

**int**speed=50;

}

**class** Bike4 **extends**javalab032{

**int**speed=100;

**void**display(){

System.***out***.println(**super**.speed);//will print speed of Vehicle now

}

**publicstaticvoid**main(String args[]){

Bike4 b=**new** Bike4();

b.display();

}

}

Output:

100

50

Ex:33**SUPER-PARENT CLASS CONSTRUCTOR**

**package** javaset2;

**publicclass** javalab033 {

javalab033(){System.***out***.println("Vehicle is created");}

}

**class** Bike5 **extends** javalab033{

Bike5(){

**super**();//will invoke parent class constructor

System.***out***.println("Bike is created");

}

**publicstaticvoid**main(String args[]){

Bike5 b=**new** Bike5();

}

}

Output:

Vehicle is created

Bike is created

Ex:34 **SUPER-INVOKES PARENT CLASS METHOD:**

**package** javaset2;

**publicclass** javalab034 {

**void** message(){System.***out***.println("welcome");}

}

**class** Student16 **extends**javalab034{

**void**message(){System.***out***.println("welcome to java");}

**void**display(){

message();//will invoke current class message() method

**super**.message();//will invoke parent class message() method

}

**publicstaticvoid**main(String args[]){

Student16 s=**new** Student16();

s.display();

}

}

Output**:**

welcome

welcome to java

ex:35 **JAVA FINAL VARIABLE**

**package** javaset2;

**publicclass** javalab035 {

final**int**speedlimit=90;//final variable

**void**run(){

speedlimit=400;

}

**publicstaticvoid**main(String args[]){

javalab035 obj=**new** javalab035();

obj.run();

}

}

Output:

Compile Time Error

Ex:36 **JAVA FINAL METHOD**

**package** javaset2;

**publicclass** javalab036 {

**finalvoid** run(){System.***out***.println("running");}

}

**class** Honda **extends** javalab036{

**void**run(){System.***out***.println("running safely with 100kmph");}

**publicstaticvoid**main(String args[]){

Honda honda= **new**Honda();

honda.run();

}

}

Output:

Compile Time Error

Ex:37 **JAVA FINAL CLASS**

**package** javaset2;

**finalclass** javalab037 {

**class** Honda1 **extends** javalab037{

**void**run(){System.***out***.println("running safely with 100kmph");}

**publicvoid**main(String args[]){

Honda honda= **new**Honda();

honda.run();

}

}

}

Output:

Compile time error

Ex:38 **IS FINAL METHOD INHERITED**:

class Honda2{

final void run(){System.out.println("running...");}

}

class Exp38 extends Honda2{

public static void main(String args[]){

new Exp38().run();

}

}

Output:

Running

Ex:39 **JAVA RUN TIME POLYMORPHISM**

class Test3{

int getRateOfInterest(){return 0;}

}

class SBI extends Test3{

int getRateOfInterest(){return 8;}

}

class ICICI extends Test3{

int getRateOfInterest(){return 7;}

}

class AXIS extends Test3{

int getRateOfInterest(){return 9;}

}

class Exp39{

public static void main(String args[]){

Test3 b1=new SBI();

Test3 b2=new ICICI();

Test3 b3=new AXIS();

System.out.println("SBI Rate of Interest: "+b1.getRateOfInterest());

System.out.println("ICICI Rate of Interest: "+b2.getRateOfInterest());

System.out.println("AXIS Rate of Interest: "+b3.getRateOfInterest());

}

}

OUTPUT:

SBI Rate of Interest: 8

ICICI Rate of Interest: 7

AXIS Rate of Interest: 9

Ex:40 **JAVA MULTILEVEL INHERITANCE**

**class** Animal{

**void** eat(){System.out.println("eating");}

}

**class** Dog **extends** Animal{

**void** eat(){System.out.println("eating fruits");}

}

**class** BabyDog **extends** Dog{

**void** eat(){System.out.println("drinking milk");}

**public** **static** **void** main(String args[]){

Animal a1,a2,a3;

a1=**new** Animal();

a2=**new** Dog();

a3=**new** BabyDog();

a1.eat();

a2.eat();

a3.eat();

}

}

OUTPUT:

eating

eating fruits

drinking Milk

EX:41 **STATIC BINDING**

**class** Dog{

**private** **void** eat(){System.out.println("dog is eating...");}

**public** **static** **void** main(String args[]){

  Dog d1=**new** Dog();

  d1.eat();

 }

}

OUTPUT:

dog is eating...

Ex:42 **DYNAMIC BINDING**

**class** Animal{

**void** eat(){System.out.println("animal is eating...");}

}

**class** Dog **extends** Animal{

**void** eat(){System.out.println("dog is eating...");}

**public** **static** **void** main(String args[]){

  Animal a=**new** Dog();

  a.eat();

 }

}

Output:

dog is eating...

Ex:43 **JAVA INSTANCEOF**

**class** Simple1{

**public** **static** **void** main(String args[]){

 Simple1 s=**new** Simple1();

 System.out.println(s **instanceof** Simple1);//true

 }

}

OUTPUT:

true

EX:44 **JAVA INSTANCEOF-II**

**class** Animal{}

**class** Dog1 **extends** Animal{//Dog inherits Animal

**public** **static** **void** main(String args[]){

 Dog1 d=**new** Dog1();

 System.out.println(d **instanceof** Animal);//true

 }

}

OUTPUT:

true

Ex:45 **ABSTRACT CLASS HAVING ABSTRACT METHOD**

**abstract** **class** Bike{

**abstract** **void** run();

}

**class** Honda4 **extends** Bike{

**void** run(){System.out.println("running safely");}

**public** **static** **void** main(String args[]){

 Bike obj = **new** Honda4();

 obj.run();

}

}

Output:

running safely

Ex:46 **ABSTRACT CLASS-CONSTRCUTOR,DATA MEMBER,METHODS**

**abstract** **class** Bike{

   Bike(){System.out.println("bike is created");}

**abstract** **void** run();

**void** changeGear(){System.out.println("gear changed");}

 }

**class** Honda **extends** Bike{

**void** run(){System.out.println("running safely..");}

 }

**class** TestAbstraction2{

**public** **static** **void** main(String args[]){

  Bike obj = **new** Honda();

  obj.run();

  obj.changeGear();

 }

}

Output:

bike is created

running safely..

gear changed

Ex:47 **JAVA INTERFACE**

**interface** printable{

**void** print();

}

**class** A6 **implements** printable{

**public** **void** print(){System.out.println("Hello");}

**public** **static** **void** main(String args[]){

A6 obj = **new** A6();

obj.print();

 }

}

OUTPUT:

Hello

Ex:48  **MULTIPLE INHERITANCE IS POSSIBLE IN JAVA INTERFACE**

**interface** Printable{

**void** print();

}

**interface** Showable{

**void** show();

}

**class** A7 **implements** Printable,Showable{

**public** **void** print(){System.out.println("Hello");}

**public** **void** show(){System.out.println("Welcome");}

**public** **static** **void** main(String args[]){

A7 obj = **new** A7();

obj.print();

obj.show();

 }

}

OUTPUT:

Hello

Welcome

Ex:49 **ABSTRACT CLASS AND INTERFACE IN JAVA**

**interface** A{

**void** a();//bydefault, public and abstract

**void** b();

**void** c();

**void** d();

}

**abstract** **class** B **implements** A{

**public** **void** c(){System.out.println("I am C");}

}

**class** M **extends** B{

**public** **void** a(){System.out.println("I am a");}

**public** **void** b(){System.out.println("I am b");}

**public** **void** d(){System.out.println("I am d");}

}

**class** Test5{

**public** **static** **void** main(String args[]){

A a=**new** M();

a.a();

a.b();

a.c();

a.d();

}}

Output:

I am a

I am b

I am c

I am d

Ex:50 **PACKAGE-IMPORT THE PACKAGE**

**package** pack;

**public** **class** A{

**public** **void** msg(){System.out.println("Hello");}

}

**package** mypack;

**import** pack.\*;

**class** B{

**public** **static** **void** main(String args[]){

   A obj = **new** A();

   obj.msg();

  }

}

Output:

Hello

Ex:51 **PRIVATE ACCESS MODIFIER**

**class** A{

**private** **int** data=40;

**private** **void** msg(){System.out.println("Hello java");}

}

**public** **class** Simple{

**public** **static** **void** main(String args[]){

   A obj=**new** A();

   System.out.println(obj.data);

   obj.msg();

   }

}

OUTPUT:

Compile Time Error

Ex:52 **DEFAULT ACCESS MODIFIER**

**package** pack;

**class** A{

**void** msg(){System.out.println("Hello");}

}

//save by B.java

**package** mypack;

**import** pack.\*;

**class** B{

**public** **static** **void** main(String args[]){

   A obj = **new** A();

   obj.msg();

  }

}

Output:

Compile Time Error

Ex:53 **PROTECTED ACCESS MODIFIER**

**package** pack;

**public** **class** A{

**protected** **void** msg(){System.out.println("Hello");}

}

**package** mypack;

**import** pack.\*;

**class** B **extends** A{

**public** **static** **void** main(String args[]){

   B obj = **new** B();

   obj.msg();

  }

}

Output:

Hello

Ex:54 **PUBLIC ACCESS MODIFIER**

**package** pack;

**public** **class** A{

**public** **void** msg(){System.out.println("Hello");}

}

**package** mypack;

**import** pack.\*;

**class** B{

**public** **static** **void** main(String args[]){

   A obj = **new** A();

   obj.msg();

  }

}

Output:

Hello

Ex:55 **ENCAPSULATION IN JAVA**

**package** com.javatpoint;

**public** **class** Student{

**private** String name;

**public** String getName(){

**return** name;

}

**public** **void** setName(String name){

**this**.name=name

}

}

//save as Test.java

**package** com.javatpoint;

**class** Test{

**public** **static** **void** main(String[] args){

Student s=**new** Student();

s.setName("DHONI");

System.out.println(s.getName());

}

}

OUTPUT:

DHONI

Ex:56 **OBJECT CLONING**

**class** Student18 **implements** Cloneable{

**int** rollno;

String name;

Student18(**int** rollno,String name){

**this**.rollno=rollno;

**this**.name=name;

}

**public** Object clone()**throws** CloneNotSupportedException{

**return** **super**.clone();

}

**public** **static** **void** main(String args[]){

**try**{

Student18 s1=**new** Student18(101,"regi");

Student18 s2=(Student18)s1.clone();

System.out.println(s1.rollno+" "+s1.name);

System.out.println(s2.rollno+" "+s2.name);

}**catch**(CloneNotSupportedException c){}

}

}

Output:

101 regi

101 regi

Ex:57 **SINGLE DIMENSIONAL JAVA ARRAY**

**class** Testarray{

**public** **static** **void** main(String args[]){

**int** a[]=**new** **int**[5];

a[0]=10;

a[1]=20;

a[2]=70;

a[3]=40;

a[4]=50;

**for**(**int** i=0;i<a.length;i++)

System.out.println(a[i]);

}}

Output:

10

20

70

40

50

Ex:58 **PASSING ARRAY TO METHOD IN JAVA**

**class** Testarray2{

**static** **void** min(**int** arr[]){

**int** min=arr[0];

**for**(**int** i=1;i<arr.length;i++)

**if**(min>arr[i])

  min=arr[i];

System.out.println(min);

}

**public** **static** **void** main(String args[]){

**int** a[]={33,3,4,5};

min(a);

}}

Output:

3

Ex:59 **MULTIDIMENSIONAL JAVAARRAY**

**class** Testarray3{

**public** **static** **void** main(String args[]){

**int** arr[][]={{1,2,3},{2,4,5},{4,4,5}};

**for**(**int** i=0;i<3;i++){

**for**(**int** j=0;j<3;j++){

   System.out.print(arr[i][j]+" ");

 }

 System.out.println();

}

}}

Output:

1 2 3

2 4 5

4 4 5

Ex:60 **COPYING A JAVA ARRAY**

**class** TestArrayCopyDemo {

**public** **static** **void** main(String[] args) {

**char**[] copyFrom = { 'd', 'e', 'c', 'a', 'f', 'f', 'e',

                'i', 'n', 'a', 't', 'e', 'd' };

**char**[] copyTo = **new** **char**[7];

        System.arraycopy(copyFrom, 2, copyTo, 0, 7);

        System.out.println(**new** String(copyTo));

    }

}

Output:

Caffein

Ex:61 **ADDITION OF 2 MATRICES IN JAVA**

**class** matadd{

**public** **static** **void** main(String args[]){

**int** a[][]={{1,3,4},{3,4,5}};

**int** b[][]={{1,3,4},{3,4,5}};

**int** c[][]=**new** **int**[2][3];

**for**(**int** i=0;i<2;i++){

**for**(**int** j=0;j<3;j++){

c[i][j]=a[i][j]+b[i][j];

System.out.print(c[i][j]+" ");

}

System.out.println();

}

}}

Output:

2 6 8

6 8 10

Ex:62 **WRAPPER CLASS EXAMPLE:PRIMITVE TO WRAPPER**

**public** **class** WrapperExample1{

**public** **static** **void** main(String args[]){

**int** a=20;

Integer i=Integer.valueOf(a);

Integer j=a;

System.out.println(a+" "+i+" "+j);

}}

OUTPUT:

20 20 20

Ex:63 **WRAPPER CLASS EXAMPLE:WRAPPER TO PRIMITIVE**

**public** **class** WrapperExample2{

**public** **static** **void** main(String args[]){

Integer a=**new** Integer(3);

**int** i=a.intValue();

**int** j=a;

System.out.println(a+" "+i+" "+j);

}}

Output:

3 3 3

Ex:64 **CALL BY VALUE IN JAVA**

**class** Operation{

**int** data=50;

**void** change(**int** data){

 data=data+100;//changes will be in the local variable only

 }

**public** **static** **void** main(String args[]){

   Operation op=**new** Operation();

   System.out.println("before change "+op.data);

   op.change(500);

   System.out.println("after change "+op.data);

 }

}

Output:

before change 50

after change 50

Ex:65 **COMMAND LINE ARGUMENT**

**class** A{

**public** **static** **void** main(String args[]){

**for**(**int** i=0;i<args.length;i++)

System.out.println(args[i]);

}

}

compile by > javac A.java

run by > java A sonoo jaiswal 1 3 abc

Output:

sonoo

jaiswal

1

3

abc

Ex:66 **JAVA STRING**

**public** **class** StringExample{

**public** **static** **void** main(String args[]){

String s1="java";//creating string by java string literal

**char** ch[]={'s','t','r','i','n','g','s'};

String s2=**new** String(ch);//converting char array to string

String s3=**new** String("example");//creating java string by new keyword

System.out.println(s1);

System.out.println(s2);

System.out.println(s3);

}}

OUTPUT:

java

strings

example

Ex:67 **IMMUTABLE STRING**

**class** Testimmutablestring{

**public** **static** **void** main(String args[]){

   String s="Sachin";

   s.concat(" Tendulkar");//concat() method appends the string at the end

   System.out.println(s);//will print Sachin because strings are immutable objects

 }

}

Output:

Sachin

Ex:68 **EQUALS()**

**class** Teststringcomparison1{

**public** **static** **void** main(String args[]){

   String s1="Sachin";

   String s2="Sachin";

   String s3=**new** String("Sachin");

   String s4="Saurav";

   System.out.println(s1.equals(s2));//true

   System.out.println(s1.equals(s3));//true

   System.out.println(s1.equals(s4));//false

 }

}

Output:

true

true

false

Ex:69 = **= OPERATOR**

**class** Teststringcomparison3{

**public** **static** **void** main(String args[]){

   String s1="Sachin";

   String s2="Sachin";

   String s3=**new** String("Sachin");

   System.out.println(s1==s2);//true (because both refer to same instance)

   System.out.println(s1==s3);//false(because s3 refers to instance created in nonpool)

 }

}

Output:

true

false

Ex:70 **compareTo()**

**class** Teststringcomparison4{

**public** **static** **void** main(String args[]){

   String s1="Sachin";

   String s2="Sachin";

   String s3="Ratan";

   System.out.println(s1.compareTo(s2));//0

   System.out.println(s1.compareTo(s3));//1(because s1>s3)

   System.out.println(s3.compareTo(s1));//-1(because s3 < s1 )

 }

}

Output:

0

1

-1

Ex:71 **STRING CONCATENATION BY +**

**class** TestStringConcatenation1{

**public** **static** **void** main(String args[]){

   String s="REGI"+"ESWARAN";

   System.out.println(s);

 }

}

Output:

REGI ESWARAN

Ex:72 **concat()**

**class** TestStringConcatenation2{

**public** **static** **void** main(String args[]){

   String s=50+30+"Dhoni"+40+40;

   System.out.println(s);

 }

}

OUTPUT:

80Dhoni4040

Ex:73 **SUBSTRING**

**public** **class** TestSubstring{

**public** **static** **void** main(String args[]){

   String s="SachinTendulkar";

   System.out.println(s.substring(6));

   System.out.println(s.substring(0,6));

 }

}

OUTPUT:

Tendulkar

Sachin

Ex:74 **toUpperCase() and toLowerCase()**

**public** **class** str{

**public** **static** **void** main(String args[]){

String s1="hello string";

String s2="JAVATPOINT”;

String s1upper=s1.toUpperCase();

String s2lower=s1.toLowerCase();

System.out.println(s1upper);

 System.out.println(s2lower);

}}

Output:

HELLO STRING

javatpoint

Ex:75 **trim()**

**public** **class** StringTrimExample{

**public** **static** **void** main(String args[]){

String s1="  hello string   ";

System.out.println(s1+"javatpoint");//without trim()

System.out.println(s1.trim()+"javatpoint");//with trim()

}}

OUTPUT:

hello string javatpoint

hello stringjavatpoint

Ex:76 **startsWith() and endsWith()**

**public** **class** StartsWithExample{

**public** **static** **void** main(String args[]){

String s="regi";

System.out.println(s.startsWith("ka"));

System.out.println(s.endsWith("sh"));

}}

Output:

true

true

Ex:77 **charAt()**

**public** **class** CharAtExample{

**public** **static** **void** main(String args[]){

String name="regi";

**char** ch=name.charAt(6);

System.out.println(ch);

}}

Output:

s

Ex:78 **length()**

**public** **class** LengthExample{

**public** **static** **void** main(String args[]){

String s1="regi";

String s2="deepikha";

System.out.println("string length is: "+s1.length());

System.out.println("string length is: "+s2.length());

}}

OUTPUT:

string length is: 7

string length is: 8

Ex:79 **intern()**

**public** **class** InternExample{

**public** **static** **void** main(String args[]){

String s1=**new** String("hello");

String s2="hello";

String s3=s1.intern();

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

System.out.println(s2==s3);

}}

OUTPUT:

false

true

Ex:80 **valueOf()**

**public** **class** StringValueOfExample{

**public** **static** **void** main(String args[]){

**int** value=30;

String s1=String.valueOf(value);

System.out.println(s1+10);

}}

Output:

3010

Ex:81 **replace()**

**public** **class** ReplaceExample1{

**public** **static** **void** main(String args[]){

String s1="regi";

String replaceString=s1.replace('a','g');

System.out.println(replaceString);

}}

OUTPUT:

Kgilgsh

Ex:82 **append()**

**class** StringBuilderExample{

**public** **static** **void** main(String args[]){

StringBuilder sb=**new** StringBuilder("Hello ");

sb.append("Java");

System.out.println(sb);//prints Hello Java

}

}

OUTPUT:

Hello Java

Ex:83 **insert()**

**class** StringBuilderExample2{

**public** **static** **void** main(String args[]){

StringBuilder sb=**new** StringBuilder("Hello ");

sb.insert(1,"Java");

System.out.println(sb);

}

}

OUTPUT:

HJavaello

Ex:84 **replace()**

**class** StringBuilderExample3{

**public** **static** **void** main(String args[]){

StringBuilder sb=**new** StringBuilder("Hello");

sb.replace(1,3,"Java");

System.out.println(sb);//prints HJavalo

}

}

OUTPUT:

HJavalo

Ex:85 **delete()**

**class** StringBuilderExample4{

**public** **static** **void** main(String args[]){

StringBuilder sb=**new** StringBuilder("Hello");

sb.delete(1,3);

System.out.println(sb);//prints Hlo

}

}

OUTPUT:

Hlo

Ex:86 **reverse()**

**class** StringBuilderExample5{

**public** **static** **void** main(String args[]){

StringBuilder sb=**new** StringBuilder(“regi");

sb.reverse();

System.out.println(sb);//prints olleH

}

}

OUTPUT:

hsaliak

Ex:87 **capacity()**

**class** StringBuilderExample6{

**public** **static** **void** main(String args[]){

StringBuilder sb=**new** StringBuilder();

System.out.println(sb.capacity());//default 16

sb.append("Hello");

System.out.println(sb.capacity());//now 16

sb.append("java is my favourite language");

System.out.println(sb.capacity());//now (16\*2)+2=34 i.e (oldcapacity\*2)+2

}

}

OUTPUT:

16

16

34

Ex:88 **ensureCapacity()**

**class** StringBuilderExample7{

**public** **static** **void** main(String args[]){

StringBuilder sb=**new** StringBuilder();

System.out.println(sb.capacity());//default 16

sb.append("Hello");

System.out.println(sb.capacity());//now 16

sb.append("java is my favourite language");

System.out.println(sb.capacity());//now (16\*2)+2=34 i.e (oldcapacity\*2)+2

sb.ensureCapacity(10);//now no change

System.out.println(sb.capacity());//now 34

sb.ensureCapacity(50);//now (34\*2)+2

System.out.println(sb.capacity());//now 70

}

}

OUTPUT:

16

16

34

34

70

Ex:89 **WITHOUT toString()**

**class** Student{

**int** rollno;

 String name;

 String city;

 Student(**int** rollno, String name, String city){

**this**.rollno=rollno;

**this**.name=name;

**this**.city=city;

 }

**public** **static** **void** main(String args[]){

   Student s1=**new** Student(101,"Raj","lucknow");

   Student s2=**new** Student(102,"Vijay","ghaziabad");

   System.out.println(s1);//compiler writes here s1.toString()

   System.out.println(s2);//compiler writes here s2.toString()

 }

}

OUTPUT:

Student@1fee6fc

Student@1eed786

Ex:90 **USING toString()**

**class** Student{

**int** rollno;

 String name;

 String city;

 Student(**int** rollno, String name, String city){

**this**.rollno=rollno;

**this**.name=name;

**this**.city=city;

 }

**public** String toString(){//overriding the toString() method

**return** rollno+" "+name+" "+city;

 }

**public** **static** **void** main(String args[]){

   Student s1=**new** Student(101,"regi","chennai");

   Student s2=**new** Student(102,"deepikha","kochi");

   System.out.println(s1);//compiler writes here s1.toString()

   System.out.println(s2);//compiler writes here s2.toString()

 }

}

Output:

101 regichennai

102 regi Chennai

Ex:91 **StringTokenizer**

**import** java.util.StringTokenizer;

**public** **class** Simple{

**public** **static** **void** main(String args[]){

   StringTokenizer st = **new** StringTokenizer("my name is regi"," ");

**while** (st.hasMoreTokens()) {

         System.out.println(st.nextToken());

     }

   }

}

Output:

my

name

is

regi

Ex:92 **JAVA REGULAR EXPRESSIONS**

**import** java.util.regex.\*;

**public** **class** RegexExample1{

**public** **static** **void** main(String args[]){

Pattern p = Pattern.compile(".s");//. represents single character

Matcher m = p.matcher("as");

**boolean** b = m.matches();

**boolean** b2=Pattern.compile(".s").matcher("as").matches();

**boolean** b3 = Pattern.matches(".s", "as");

System.out.println(b+" "+b2+" "+b3);

}}

OUTPUT:

true truetrue

Ex:93 **THE .(DOT)**

import java.util.regex.\*;

class RegexExample2{

public static void main(String args[]){

System.out.println(Pattern.matches(".s", "as"));//true (2nd char is s)

System.out.println(Pattern.matches(".s", "mk"));//false (2nd char is not s)

System.out.println(Pattern.matches(".s", "mst"));//false (has more than 2 char)

System.out.println(Pattern.matches(".s", "amms"));//false (has more than 2 char)

System.out.println(Pattern.matches("..s", "mas"));//true (3rd char is s)

}}

OUTPUT:

Compile by: javac RegexExample2.java

Run by: java RegexExample2

true  
false  
false  
false  
true

Ex:94 **REGEX CHARACTER CLASSES**

**import** java.util.regex.\*;

**class** RegexExample3{

**public** **static** **void** main(String args[]){

System.out.println(Pattern.matches("[amn]", "abcd"));

System.out.println(Pattern.matches("[amn]", "a"));

System.out.println(Pattern.matches("[amn]", "ammmna"));

}}

OUTPUT:

false  
true  
false

Ex:95 **REGEX QUANTIFIERS**

**import** java.util.regex.\*;

**class** RegexExample4{

**public** **static** **void** main(String args[]){

System.out.println("? quantifier ....");

System.out.println(Pattern.matches("[amn]?", "a"));

System.out.println(Pattern.matches("[amn]?", "aaa"));

System.out.println(Pattern.matches("[amn]?", "aammmnn"));

System.out.println(Pattern.matches("[amn]?", "aazzta"));

System.out.println(Pattern.matches("[amn]?", "am"));

System.out.println("+ quantifier ....");

System.out.println(Pattern.matches("[amn]+", "a"));

System.out.println(Pattern.matches("[amn]+", "aaa"));

System.out.println(Pattern.matches("[amn]+", "aammmnn"));

System.out.println(Pattern.matches("[amn]+", "aazzta"));

System.out.println("\* quantifier ....");

System.out.println(Pattern.matches("[amn]\*", "ammmna"));

}}

OUTPUT:

? quantifier ....  
true  
false  
false  
false  
false  
+ quantifier ....  
true  
true  
true  
false  
\* quantifier ....  
true

Ex:96 **REGEX METACHARACTERS**

**import** java.util.regex.\*;

**class** RegexExample5{

**public** **static** **void** main(String args[]){

System.out.println("metacharacters d....");\\d means digit

  System.out.println(Pattern.matches("\\d", "abc"));

System.out.println(Pattern.matches("\\d", "1"));

System.out.println(Pattern.matches("\\d", "4443"));

System.out.println(Pattern.matches("\\d", "323abc"));

  System.out.println("metacharacters D....");\\D means non-digit

  System.out.println(Pattern.matches("\\D", "abc"));

System.out.println(Pattern.matches("\\D", "1"));

System.out.println(Pattern.matches("\\D", "4443");

System.out.println(Pattern.matches("\\D", "323abc"));

System.out.println(Pattern.matches("\\D", "m"));

  System.out.println("metacharacters D with quantifier....");

System.out.println(Pattern.matches("\\D\*", "mak"));

  }}

OUTPUT:

False

True

False

False

False

False

False

False

True

True

Ex:97 **REGULAR EXPRESSION QUESTION1**

**import** java.util.regex.\*;

**class** RegexExample6{

**public** **static** **void** main(String args[]){

System.out.println(Pattern.matches("[a-zA-Z0-9]{6}", "arun32"));

System.out.println(Pattern.matches("[a-zA-Z09]{6}", "kkvarun32"));

System.out.println(Pattern.matches("[a-zA-Z0-9]{6}", "JA2Uk2"));

System.out.println(Pattern.matches("[a-zA-Z0-9]{6}", "arun$2"));

}}

OUTPUT:

true  
false  
true  
false

Ex:98 **REGULAR EXPRESSION QUESTION2**

**import** java.util.regex.\*;

**class** RegexExample7{

**public** **static** **void** main(String args[]){

System.out.println("by character classes and quantifiers ...");

System.out.println(Pattern.matches("[789]{1}[0-9]{9}", "9953038949"));

System.out.println(Pattern.matches("[789][0-9]{9}", "9953038949"));

System.out.println(Pattern.matches("[789][0-9]{9}", "99530389490"));

System.out.println(Pattern.matches("[789][0-9]{9}", "6953038949"));

System.out.println(Pattern.matches("[789][0-9]{9}", "8853038949"));

System.out.println("by metacharacters ...");

System.out.println(Pattern.matches("[789]{1}\\d{9}", "8853038949"));

System.out.println(Pattern.matches("[789]{1}\\d{9}", "3853038949"));

}}

OUTPUT:

by character classes and quantifiers ...  
true  
true  
false  
false  
true  
by metacharacters ...  
true  
false