|  |
| --- |
| 1. Steps to install java |
| Download "Java development kit 8” from Oracle site, extract the file, Install the open source file, Open command prompt and enter command “javac” and verify whether java is installed successfully or not. |
| 1. Steps to install eclipse |
| 1)Download the eclipse file from any oracle website 2) extract the folder 3)Click on install |
| 1. Steps to create workspace |
| File> Search workspace> Create workspace by its name |
|  |
| 1. Steps to create project |
| File> new> Project> Select “Java Project”> Enter project name. |
| 1. create .java file/class |
| File> new> class file> Class name should always begin with Upper case |
|  |
| 1. how to create packages and what is best way to give name |
| File> new> package> Packages should always begin with lower case letters |
| 1. what is main method will do? |
| Main method is the point where Java program starts executing. |
| 1. creating property/data members : |
|  |
| 1. what is data type and different data types` |
| Data types are the one which defines the variable, there are eight primitive data types, they are Int, byte , long, short, Boolean, char, double, float |
|  |
| 1. What is variable? |
| Variable is a name of memory location. There are three types of variables in java: local, instance and static. |
|  |
| 1. creating method with void |
| public void pup() {  System.out.println("Puppy");  } |
| 1. creating variable, we can create variables inside method |
| public class Test {  public void pupAge() {  int age = 0;  age = age + 7;  System.out.println("Puppy age is : " + age);  }  } |
| 1. creating method with return data type, we can return int/string/double/float/date etc |
| Public int add(){  Int a=10;  Int b=30;  Int c=a+b;  Return c;  } |
| **Note: value that we specify after return keyword should be of data type that is specified in method signature** |
|  |
| 1. method that will return hard coded value |
| Hard coding is not only part of java. It actually comes when we put the original variables and data values In place of hard coding |
|  |
| 1. method that will return property value |
|  |
|  |
| 1. creating method with return data type and parameter |
|  |
|  |
| 1. creating static property: |
| The static property in java is used for memory management mainly. We can apply java static keyword with variables, methods, blocks and nested class. The static keyword belongs to the class than instance of the class. The static can be variable |
|  |
| 1. creating static method |
| Public class Test  {  Public static int add(int a, int b);  {  Int c= (a+b);  System.out.println(“Result:” +c);  }  Public static void main(strings[] args);  {  Test t=new Test;  t.add(10,20);  }  } |
| 1. creating object |
| Class Test  {  Void navika()  {  System.out.println(“Method M1”);  }  Public static void main(string[] args);  {  Test t= new test();// creating object for class Test  t.navika();  }  } |
| 1. calling method with void |
| Class Test  {  public void pup()  {  System.out.println("Puppy");  }  Public static void main(string[] args);  {  Test t= new test();// creating object for class Test  t.pup();  }  } |
| 1. calling method with no return and parameter |
| Class Test  {  Public m1()  {  System.out.println(“Method m1”);  }  }  Public static void main(string[] args);  {  Test t= new test();// creating object for class Test  t.m1();  } |
| 1. calling method with return and no parameter |
| Public class Test  {  Public int total();  {  int a\_value =10;  return a\_value;  }  Public static void main(string[] args);  {  Test t=new Test;  t.total();  }  } |
| 1. calling method with return and parameter   public class ExampleMinNumber{  public static void main(String[] args){  int a=11;  int b=6;  int c=minFunction(a,b);  System.out.println(“Minimum Value=”+c);  }  Public static int minFunction(int n1, int n2){  Int min;  If(n1>n2)  Min=n2;  Else  Min=n1;  Return min;  }  } |
| 1. calling method with return and string the return data |
| public class PerfectNumber {  public boolean PerfectNumber(int number){  int temp=2;  for(int i=1;i&lt;=number/2;i++){  if(number%1==0){  temp+=i;  }  }  if(temp==number){  System.out.println(&quot;It is a perfect number&quot;);  return true;  }  else{  System.out.println(&quot;It is not a perfect number&quot;);  return false;  }  //return value;  } |
| 1. calling static method |
|  |
| 1. using static property: it will maintain |
|  |
| **Ans: public static int empid;** |
|  |
| 1. create classes under multiple packages |
| Under single java project, we can create multiple packages and multiple classes. |
| 1. calling classes under different packages |
| If access modifier for a class is Public, then it can be called by creating an object to it. |
| 1. write code to handle exceptions with try/catch/finally |
| **package** exception;  **public** **class** ExceptionDemo {  **public** **static** **void** hello(){  **try**{  System.***out***.println("hi");  **return**;  }  **catch**(RuntimeException e)  {  }  **finally**{  System.***out***.println("finally");  }  }  **public** **static** **void** main(String[] args){  *hello*();  }  } |
| 1. what is final keyword, create final class, final method, final property |
| Final is a keyword which is applicable for classes, methods and variables.  If a class is declared as final, then that class cannot be extended i.e we cannot create child class.  If a method is declared as final, then that method cannot be overridden in child.  If a variable is declared as final, then that variable is constant. |
| 1. write code for interface and create class to implement that interface |
| **package** interfac;  **public** **interface** Message{  **public** **void** morn();  **public** **void** even();  **public** **void** gn();  }  **package** interfac;  **public** **class** TestClient **implements** Message  {  **public** **static** **void** main(String[] args) {  TestClient t=**new** TestClient();  t.morn();  t.even();  t.gn();  }  **public** **void** morn() {  System.***out***.println("Good morning");    }  **public** **void** even() {  System.***out***.println("Good evening");    }  **public** **void** gn() {  System.***out***.println("Good Night");    }  } |
| 1. write code for creating abstract class |
| **package** Abstraction;  **public** **abstract** **class** TestC {  **abstract** **void** m1();  **abstract** **void** m2();  **abstract** **void** m3();  **abstract** **void** m4();  **void** m5(){  System.***out***.println("Method m5()");  }  }  **public** **class** TestE **extends** TestC{  **void** m1(){  System.***out***.println("Method m1()");  }  **void** m2(){  System.***out***.println("Method m2()");  }  **void** m3(){  System.***out***.println("Method m3()");  }  **void** m4(){  System.***out***.println("Method m4()");  }  **public** **static** **void** main(String[] args) {  TestE t= **new** TestE();  t.m1();  t.m2();  t.m3();  t.m4();  t.m5();  }  } |
| 1. implement method overloading |
| **public** **class** Test1 {  //Method overloading  //Class contains more than 1method with same name, different no of args and same data types (or) same no of args and different data types  **void** m1(**int** a, **int** b)  {  System.***out***.println("Method 1");  }  **void** m1(**char** c, **int** a)  {  System.***out***.println("Method 2");  }  **void** m1(**int** a, **int** b, **int** c)  {  System.***out***.println("Method 3");  }  **public** **static** **void** main(String[] args)  {  Test1 t= **new** Test1();  t.m1(10, 20);  t.m1('a', 10);  t.m1(10, 20, 25);  }  } |
| implement method overriding  **package** overriding;  **public** **class** Parent {  **void** Rita(){  System.***out***.println("Rita is a singer");  }  }  **public** **class** Child **extends** Parent{    **void** Rita()  {  System.***out***.println("Rita is a Dancer");  }  **public** **static** **void** main(String[] args) {    Child c= **new** Child();  c.Rita();  }  } |
| 1. implementing polymorphism |
| 1.Method Overloading  2.Method Overriding |
| 1. implementing interface |
| **package** interfac;  **public** **interface** Message{  **public** **void** morn();  **public** **void** even();  **public** **void** gn();  }  **package** interfac;  **public** **class** TestClient **implements** Message  {  **public** **static** **void** main(String[] args) {  TestClient t=**new** TestClient();  t.morn();  t.even();  t.gn();  }  **public** **void** morn() {  System.***out***.println("Good morning");    }  **public** **void** even() {  System.***out***.println("Good evening");    }  **public** **void** gn() {  System.***out***.println("Good Night");    }  } |
| 1. How to do inheritance in java (using extend keyword) |
| **package** inheritance;  **public** **class** TestA {  **void** m1(){  System.***out***.println("Parent class method m1()");  }  }  **package** inheritance;  **public** **class** TestB **extends** TestA{  **void** m1(){  System.***out***.println("Child class method m1()");  }  **void** m2(){  **this**.m1(); //current class method m1. i.e child class m1method  **super**.m1(); //super class method m1. i.e parent class m1method  }  **public** **static** **void** main(String[] args) {    TestB t= **new** TestB();  t.m1();  t.m2();  }  } |
| 1. write code to add items to integer, string array |
| public static void main(String[] args) {  int[] series = new int[0];  int x = 5;  series = addInt(series, x);  System.out.print(&quot;New series: &quot;);  for (int i = 0; i &lt; series.length; i++){  if (i == series.length - 1){  System.out.println(series[i]);  }  else{  System.out.print(series[i] + &quot;, &quot;);  }  }  } |
| 1. write code to retrieve items from integer, string array |
| public int getArrayIndex(int[] arr,int value) {  int k=0;  for(int i=0;i&lt;arr.length;i++){  if(arr[i]==value){  k=i;  break;  }  }  return k;  } |
| 1. write code to add items to ArrayList collection |
| import java.util.\* ;  public class ArrayList  {  public static void main ( String[] args)  {  ArrayList&lt;String&gt; names = new ArrayList&lt;String&gt;();  System.out.println(&quot;initial size: &quot; + names.size() );  names.add(&quot;prasu&quot;);  names.add(&quot;vasu&quot;);  names.add(&quot;bhavana&quot;);  System.out.println(&quot;new size: &quot; + names.size() );  for ( int j=0; j&lt;names.size(); j++ )  System.out.println(&quot;element &quot; + j + &quot;: &quot; + names.get(j) );  }  } |
| 1. write code to retrieve items from arraylist (using for each loop\_ |
| private void Customer Info form (object sender, EventArgs e)  {  ArrayList arrayList = new ArrayList();  arrayList.Add(&quot;Customer1&quot;);  arrayList.Add(&quot;Customer2&quot;);  arrayList.Add(&quot;Customer3&quot;);  arrayList.Add(&quot;Customer4&quot;);  arrayList.Add(&quot;Customer5&quot;);  string str = string.Empty;  foreach (string strName in arrayList)  {  str += strName + &quot;\n&quot;;  } |
| 1. write code to add items HashMap. Write code to retrieve items HashMap |
|  |
| **package** hashmap;  **import** java.util.\*;  **public** **class** HashMapDemo {  **public** **static** **void** main(String args[]) {  // create hash map  HashMap newmap = **new** HashMap();    // populate hash map  newmap.put(1, "tutorials");  newmap.put(2, "point");  newmap.put(3, "is best");    // get value of key 3  String val=(String)newmap.get(3);    // check the value  System.***out***.println("Value for key 3 is: " + val);  }  } |
| 1. Write code to add items to hashset. Write code to retrieve items to hashset |
| import java.util.\*;  public class HashSetDemo {  public static void main(String args[]) {  // create a hash set  HashSet hs = new HashSet();    // add elements to the hash set  hs.add("B");  hs.add("A");  hs.add("D");  hs.add("E");  hs.add("F");  hs.add("F");  System.out.println(hs);  }  }  Output:  [D, E, F, A, B] |
| 1. write code to connect to JDBC to get rows from employee table |
| String selectTableSQL = &quot;SELECT USER\_ID, USERNAME from DBUSER&quot;  Statement statement = dbConnection.createStatement();  ResultSet rs = statement.executeQuery(selectTableSQL);  while (rs.next()) {  String userid = rs.getString(&quot;USER\_ID&quot;);  String username = rs.getString(&quot;USERNAME&quot;);  } |
| 1. Write method to return list of rows code to loop throughs |
| 1. create Employee class |
| 1. Add employee class to list collection |
| 1. create method that return list of employee collection |
| **package** employeedetails;  **import** java.util.ArrayList;  **import** java.util.List;    **public** **class** EmployeeClass {    **int** empID;  String empName;  **int** empAge;    **public** EmployeeClass(**int** empID,String empName,**int** empAge)  {  **this**.empID=empID;  **this**.empName=empName;  **this**.empAge=empAge;    }  **public** **int** getEmpId() {  **return** empID;  }  **public** **void** setEmpid(**int** empid) {  **this**.empID = empid;  }  **public** String getname() {  **return** empName;  }  **public** **void** setname(String name) {  **this**.empName = name;    }  **public** **int** getEmpAge() {  **return** empAge;  }  **public** **void** setEmpAge(**int** empAge) {  **this**.empAge = empAge;  }    }  **class** EmployeeTester {    **public** **static** **void** main(String[] args) {    List<EmployeeClass> list=**new** ArrayList<EmployeeClass>();    list.add(**new** EmployeeClass(12813,"Emp1",23));  list.add(**new** EmployeeClass(12614,"Emp2",50));  list.add(**new** EmployeeClass(12820,"Emp3",30));  list.add(**new** EmployeeClass(18900,"Emp4",45));    **for**(**int** i=0;i<list.size();i++){  System.***out***.print("Empid: "+list.get(i).getEmpId()+" ");  System.***out***.println("Name: "+list.get(i).getname()+ " ");  System.***out***.println("Age: "+list.get(i).getEmpAge());  }  }  } |
| 1. Difference between string, string buffer, string builder with example   String: Stringis *immutable* (once created cannot be changed) object. |
| String Buffer: It is mutable means one can change the value of the object.  String Builder: It is also not thread safe**.** |
| 1. how to update the data into XML file and read data from XML file |

We can update using web.XML or POM.XML

<?xml version=”1.0”; encoding=”UTF-8”; standalone=”no”?>

<data>

<username>admin</username>

<password>12345</password>

<interval>1</interval>

<timeout>90</timeout>

<startdate>01/01/2013</startdate>

<enddate>06/01/2013</enddate>

<time>1110</time>

</data>