|  |
| --- |
| **1.Steps to install java** |
| 1)Download JDK  2)Install JDK  3) Extract the File  4) Install the open source file  5) Open the command prompt to and type the command JAVA. To verify whether java is installed in the respective PC or not |
| **2.Steps to install eclipse** |
| 1)Download the eclipse file from any oracle website 2) extract the folder 3)Click on install 4) Save the app in the Applications folder |
| Steps to create workspace |
| File---> Search workspace ---> Create workspace by its name |
|  |
| **3.Steps to create project** |
| File---> new---> Project----> Select type of the project |
| create .java file/class |
| File ---> new---> class file--> Java ,,,, class file should always begin with upper case letters |
|  |
| **4.How to create packages and what is best way to give name** |
| file-->new ---> package ,,,, Packages should always begin with lower case letters |
| **5.what is main method will do?** |
| Main is the point of entry for a java program, It runs and executes the program . |
|  |
|  |
|  |
| **6.what is data type and different data types`** |
| Data types are the one which defines the variable ,,, they are eight primitive data types like  1)boolean  2)int  3)char  4)double  5)long  6)float  7) short |
| **7.What is variable?** |
| Variables are the one which has piece of memory stored along with it |
| creating method with void |
| public Static void main () ,,, method created with void will not have any return value |
|  |
| **8.Creating variable, we can create variables inside method?** |
| variable can be created by associating a data type along with it we cant create a static variable inside  a method ,as it doesn't make any sense |
|  |
| **9.creating method with return data type, we can return int/string/double/float/date etc** |
| yes we can return a value from any of the specific method |
| **Note: value that we specify after return keyword should be of data type that is specified in**  **method signature** |
|  |
| **10.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 |
|  |
| **11.Create default/paramterzied constructors?** |
| when we don't declare a constructor ,, then a default constructor is created ,, when an object is declared and the values are passed in it then it is called as a paramterized constructor |
|  |
|  |
|  |
|  |
| **12.creating method with return data type and parameter?** |
| public int total(int a)  {  int b=a+10;  return b;  } |
| **13.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 |
|  |
|  |
| **14.creating static method?**  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  DecimaltoBinary dtb= **new** DecimaltoBinary();  dtb.printBinaryFormat(10);  }  } |
|  |
| **15.create static block ?**  public class Foo {  private static final int widgets = getWidgets();  static int getWidgets() {  int first = Widgets.getFirstCount();  int second = Widgets.getSecondCount();    return first + second;  }  } |
|  |
|  |
|  |
|  |
|  |
| **16.creating object?**  **public** **class** DecimaltoBinary {    **public** **void** printBinaryFormat(**int** number){  **int** binary[]= **new** **int**[10];  **int** index=4;  **while**(number>0){  binary[index++]=number%2;  number= number/2;  }  **for**(**int** i= index-1;i>=0;i--){  System.***out***.println(binary[6]);  }  }  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  DecimaltoBinary dtb= **new** DecimaltoBinary();---------🡪 Creating new object  dtb.printBinaryFormat(10);  }  } |
|  |
| **17.calling method with void ?**  **public** **static** **void** main(String[] args) {  System.out.println(“Hello world”);  }  Note: Void doesn’t have any return value. |
|  |
| **18.calling method with no return and parameter ?**  **public** **class** Swap1 {               **public** **void** swap (**int** a, **int** b)         {           **int** temp = a;           a = b;           b = temp;         }      } **class** SwapTester{       **public** **static** **void** main(String args[]){             Swap1 S = **new** Swap1();             S.swap(6,4);            System.***out***.println("successfully swapped");     }                      } |
|  |
| **19.Calling method with return and no parameter ?**  **package** saibersys; **public** **class** Add{ **int**a=1; int b=2; c=a+b; Add a+b(); } **public** **static** **void** main(String[]args) {                                            return(c); } |
|  |
| **20.Calling method with return and parameter ?**  double total(double aNumber); { double a\_Value = aNumber+40; return a\_value;} doublel(double aNumber) { double a\_Value = aNumber+60; return a\_Value; } |
|  |
| **21.write code for interface and create class to implement that interface ?**  interface Animal {  public void eat();  public void travel();  }  ///////Implementing the above interface |
| public class MammalInt implements Animal {  public void eat() {  System.out.println("Mammal eats");  }  public void travel() {  System.out.println("Mammal travels");  }  public int noOfLegs() {  return 0;  }  public static void main(String args[]) {  MammalInt m = new MammalInt();  m.eat();  m.travel();  }  } |
| **22.write code for creating abstract class ?**  public abstract class Employee {  private String name;  private String address;  private int number;  public Employee(String name, String address, int number) {  System.out.println("Constructing an Employee");  this.name = name;  this.address = address;  this.number = number;  }    public double computePay() {  System.out.println("Inside Employee computePay");  return 0.0;  }    public void mailCheck() {  System.out.println("Mailing a check to " + this.name + " " + this.address);  }  public String toString() {  return name + " " + address + " " + number;  }  public String getName() {  return name;  }    public String getAddress() {  return address;  }    public void setAddress(String newAddress) {  address = newAddress;  }    public int getNumber() {  return number;  }  }  ///// Implementing the above Abstract demo |
| public class AbstractDemo {  public static void main(String [] args) {    Employee e = new Employee("George W.", "Houston, TX", 43);  System.out.println("\n Call mailCheck using Employee reference--");  e.mailCheck();  }  } |
| **23.implement method overloading ?**  **package** test;  **public** **class** test2 {  **void** s1(**float** a,**float** b)  {  System.***out***.println("salary 1");    }  **void** s1(**int** c,**int** d)  {  System.***out***.println("salary 2");  }  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  test2 t = **new** test2();  t.s1(20.5f,30.5f);  t.s1(20,70);  }  } |
|  |
| **24.implement method overriding ?**  **package** overriding;  **public** **class** Animal {  **void** Tiger(){    System.***out***.println("Tiger is a animal");      }  }  **package** overriding;  **public** **class** Dog **extends** Animal {    **void** Tiger()  {  System.***out***.println("tiger is a good animal");  }  **public** **static** **void** main(String args[]){    Dog d=**new** Dog();  d.Tiger();    }  } |
|  |
|  |
|  |
|  |
| **25.How to do inheritance in java (using extend keyword)**  class MyBaseClass{  protected void disp()  {  System.out.println("Parent class method");  }  }  class MyChildClass extends MyBaseClass{ /////// This is the way when we use inheritance  public void disp(){  System.out.println("Child class method");  }  public static void main( String args[]) {  MyChildClass obj = new MyChildClass();  obj.disp();  }  } |
|  |
| **26.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("New series: ");  for (int i = 0; i < series.length; i++){  if (i == series.length - 1){  System.out.println(series[i]);  }  else{  System.out.print(series[i] + ", ");  }  }  } |
|  |
| **27.write code to retrieve items from integer, string array ?**  public int getArrayIndex(int[] arr,int value) {  int k=0;  for(int i=0;i<arr.length;i++){  if(arr[i]==value){  k=i;  break;  }  }  return k;  } |
|  |
| **28.write code to add items to ArrayList collection ?**  import java.util.\* ;  public class ArrayList  {  public static void main ( String[] args)  {    ArrayList<String> names = new ArrayList<String>();    System.out.println("initial size: " + names.size() );  names.add("Ram");  names.add("Krishna");  names.add("Hari");  System.out.println("new size: " + names.size() );      for ( int j=0; j<names.size(); j++ )  System.out.println("element " + j + ": " + names.get(j) );  }  } |
|  |
| **29.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("Customer1");  arrayList.Add("Customer2");  arrayList.Add("Customer3");  arrayList.Add("Customer4");  arrayList.Add("Customer5");    **string** str = **string**.Empty;  **foreach** (**string** strName **in** arrayList)  {  str += strName + "\n";  } |
|  |
| **30.write code to add items HashMap ?**  hm.put("A", new Person("p1"));  hm.put("B", new Person("p2"));  hm.put("C", new Person("p3"));  hm.put("D", new Person("p4"));  hm.put("E", new Person("p5"));  Set<Map.Entry<String, Person>> set = hm.entrySet();  for (Map.Entry<String, Person> me : set) {  System.out.println("Key :"+me.getKey() +" Name : "+ me.getValue().getName()+"Age :"+me.getValue().getAge());  } |
|  |
| **31.write code to retrieve items HashMap ?**  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);  }  } |
|  |
|  |
|  |
|  |
|  |
| **32.write code to connect to JDBC to get rows from employee table ?**  String selectTableSQL = "SELECT USER\_ID, USERNAME from DBUSER";  Statement statement = dbConnection.createStatement();  ResultSet rs = statement.executeQuery(selectTableSQL);  while (rs.next()) {  String userid = rs.getString("USER\_ID");  String username = rs.getString("USERNAME");  } |
|  |
| **33.Write method to return list of rows code to loop throughs**  **private** **void** **Customer Info form** (**object** sender, EventArgs e)  {  ArrayList arrayList = **new** ArrayList();  arrayList.Add("Customer1");  arrayList.Add("Customer2");  arrayList.Add("Customer3");  arrayList.Add("Customer4");  arrayList.Add("Customer5");    **string** str = **string**.Empty;  **foreach** (**string** strName **in** arrayList)  {  str += strName + "\n";  } |
|  |
|  |
|  |
| **create Employee class**  Go to project🡪package🡪new-🡪 class🡪name the class name as Employee |
|  |
| **34.create method that return list of employee collection**  public class Employee {  int empid;  String name;  int age;  public Employee(int empid,String name,int age)  {  this.empid=empid;  this.name=name;  this.age=age;  }  public int getEmpid() {  return empid;  }  public void setEmpid(int empid) {  this.empid = empid;  }  public String getname() {  return name;  }  public void setname(String name) {  this.name = name;  }  public int getAge() {  return age;  }  public void setAge(int age) {  this.age = age;  }  }  comparator class:  public class Employee\_comparator implements Comparator<Employee> {  @Override  public int compare(Employee object1, Employee object2) {  return object1.getname().compareTo(object2.getname());  }  } |
|  |
| **35.Difference between string, string buffer, string builder with example ?**  String :- String is a kind of primitive data type  String buffer:- Java provides the StringBuffer and String classes, and the String class is used to manipulate character **strings** that cannot be changed.  String builder :- It is faster then the string builder and it is completely thread safe |
|  |
|  |
|  |
| **36.how to update the data into XML file and read data from XML file ?**  We can update using web.XML or POM.XML  Example  <?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>  <ttime>1110</ttime>  </data> |
|  |