|  |
| --- |
| **Steps to install java** |
| 1)Download JDK 2)Install JDK 3) Extract the File 4) Install the open source file 5) Open the Terminal to and type the command JAVA. To verify whether java is installed in the respective PC or not |
| **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 |
|  |
| **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 |
|  |
| **How to create packages and what is best way to give name** |
| file---->new ----> package ,,,, Packages should always begin with lower case letters |
| **what is main method will do?** |
| Main is the point of entry for a java program ,, It runs and executes the program . |
|  |
|  |
|  |
| **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 |
| **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 |
|  |
| **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 |
|  |
| **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** |
|  |
| **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 |
|  |
| **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 |
|  |
|  |
|  |
|  |
| **creating method with return data type and parameter?** |
| public class Circle {  private int x, y, radius;  public void setOrigin(int x, int y) {    } } |
| **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 |
|  |
|  |
| **creating static method?**  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  DecimaltoBinary dtb= **new** DecimaltoBinary();  dtb.printBinaryFormat(10);  }  } |
|  |
| **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;  }  } |
|  |
|  |
|  |
|  |
|  |
| **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);  }  } |
|  |
| **calling method with void ?**  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  DecimaltoBinary dtb= **new** DecimaltoBinary();  dtb.printBinaryFormat(10);  }  } |
|  |
| **calling method with no return and parameter ?**  public double calculateAnswer(double wingSpan, int numberOfEngines,  double length, double grossTons) {  Volume == length\* grossTons  } |
|  |
| **Calling method with return and no parameter ?**    Return type<----int total( int aNumber) {  int a\_value =aNumber+10;  return a\_value;  } |
|  |
| **Calling method with return and parameter ?**  int total( int aNumber )  {  int a\_Value = aNumber + 20;  return a\_Value;  }  int total( int aNumber )  {  int a\_Value = aNumber + 50;  return a\_Value;   } |
|  |
| **Calling method with return and storing the return data ?**  **public** **class** PerfectNumber {    /\* public boolean PerfectNumber(int number){  int temp=2;  for(int i=1;i<=number/2;i++){  if(number%1==0){  temp+=i;    }  }  if(temp==number){  System.out.println("It is a perfect number");  return true;  }  else{  System.out.println("It is not a perfect number");  return false;    }  //return value;  }  \*/ |
|  |
| **Calling static method ?**  **// Question repeated** |
|  |
| **using static property: it will maintain** |
|  |
| **Ans: public static int empid;** |
|  |
| **create classes under multiple packages** |
| **class** Computer {  Computer() {  System.out.println("Constructor of Computer class.");  }    **void** computer\_method() {  System.out.println("Power gone! Shut down your PC soon...");  }    **public** **static** **void** main(String[] args) {  Computer my = **new** Computer();  Laptop your = **new** Laptop();    my.computer\_method();  your.laptop\_method();  }  }    **class** Laptop {  Laptop() {  System.out.println("Constructor of Laptop class.");  }    **void** laptop\_method() {  System.out.println("99% Battery available.");  }  } |
| **calling classes under different packages**  **Package Software**  **Public class** Computer {  Computer() {  System.out.println("Constructor of Computer class.");  }    **void** computer\_method() {  System.out.println("Power gone! Shut down your PC soon...");  }    **public** **static** **void** main(String[] args) {  Computer my = **new** Computer();  Laptop your = **new** Laptop();    my.computer\_method();  your.laptop\_method();  }  }   Package Hardware  **Public class** Laptop {  Laptop() {  System.out.println("Constructor of Laptop class.");  }    **void** laptop\_method() {  System.out.println("99% Battery available.");  }  } |
|  |
| **write code to handle exceptions with try/catch/finally?**  public class TryExample {  public static void main(String[] args)  {  try {  int[] i = {1, 2, 3};  int x = i[3];//Change to 2 to see "return" result  return;  } catch (ArrayIndexOutOfBoundsException e) {  System.out.println("caught");  System.exit(0);  } finally {  System.out.println("finally");  }  }  } |
|  |
| **what is checked exception/unchecked exception**  **Checked Exception** :- which are not forseen by the programmer,, Ex:- File not Found |
| **Unchecked Exceptions**:- Exceptions arised while running a program ,,Ex:- IO Exceptions  Run Time Exceptions |
| **what is final keyword, create final class, final method, final property** |
| Final Key word is mainly used to restrict the java variable  **Public class** Bike{  **final** **int** speedlimit=90;//final variable  **void** run(){   speedlimit=400;  }  **public** **static** **void** main(String args[]){   Bike9 obj=**new**  Bike9();  obj.run();  }  } |
| **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();  }  } |
| **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();  }  } |
| **implement method overloading ?**  class MyClass {  int height;  MyClass() {  System.out.println("bricks");  height = 0;  }  MyClass(int i) {  System.out.println("Building new House that is " + i + " feet tall");  height = i;  }  void info() {  System.out.println("House is " + height + " feet tall");  }  void info(String s) {  System.out.println(s + ": House is " + height + " feet tall");  }  }  public class MainClass {  public static void main(String[] args) {  MyClass t = new MyClass(0);  t.info();  t.info("overloaded method");    //Overloaded constructor:  new MyClass();  }  } |
|  |
| **implement method overriding ?**  class Man{  public void eat()  {  System.out.println("Human is eating");  }  }  class Boy extends Man{  public void eat(){  System.out.println("Boy is eating");  }  public static void main( String args[]) {  Boy obj = new Boy();  obj.eat();  }  } |
| **implementing polymorphism ?**  Through method overloading and over riding |
|  |
| **implementing interface ?**  Implementing interface with the Abstract class |
|  |
| **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();  }  } |
|  |
| **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] + ", ");  }  }  } |
|  |
| **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;  } |
|  |
| **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("prasu");  names.add("vasu");  names.add("bhavana");  System.out.println("new size: " + names.size() );      for ( int j=0; j<names.size(); j++ )  System.out.println("element " + j + ": " + names.get(j) );  }  } |
|  |
| **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";  } |
|  |
| **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());  } |
|  |
| **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);  }  } |
|  |
|  |
|  |
|  |
|  |
| **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");  } |
|  |
| **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 |
|  |
| **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());  }  } |
|  |
| **Difference between string, string buffer, string builder with example ?**  String :- String is a kind of primitive data type  String buffer:- Used to change the immutable string  String builder :- It is faster then the string builder and it is completely thread safe |
|  |
| **write a code to save data into excel file and read from excel file (POI and jexcel API)** |
|  |
| **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> |
|  |