1. **Steps to install java**

Download the jre-8u65-macosx-x64.pkg file.   
Review and agree to the terms of the license agreement before downloading the file.

Double-click the .pkg file to launch it

Double-click on the package icon to launch install Wizard

The Install Wizard displays the Welcome to Java installation screen. Click **Next**

After the installation has completed, a confirmation screen appears. Click **Close** to finish the installation process.

1. **Steps to install eclipse**

Go to <http://www.eclipse.org/downloads/> to download a copy of eclipse.

Select appropriate version for your OS click on ok

The download will go to your Downloads folder.

Drag the "eclipse" folder into your Applications folder.

Double click the "eclipse" folder.

You'll see an application named Eclipse

When you launch Eclipse for the first time, you'll be asked "'Eclipse' is an application downloaded from the Internet. Are you sure you want to open it?" Click "Open."

Then it opens workspace launcher, give your work space click on ok.

1. **Steps to create workspace**

When you click on Eclipse a workspace launcher window will be opens in that field browse your folder in which you want to save your files.

1. **Steps to create project**

Create a new project in eclipse: File->New -> java project-> give a name to project.

1. **create .java file/class**

Create a Class: Right click on project-> New -> Class, give a name to Class.

1. **how to create packages and what is best way to give name**

Create new Package: Right click on project -> New-> Package.

Package names are written in all lower case to avoid conflict with the names of classes or interfaces.

1. **what is main method will do?**

It is the entry point to the program. the main method is the method that will be called if you try to just run the class.

1. **creating property/data members**

int salary;

1. **what is data type and different data types**

It will represent the type of variable

int a, b, c; // Declares the integers (int), a, b, and c.

int a = 10, b = 10; // Example of initialization

byte B = 22; // initializes a byte type variable B.

double pi = 3.14159; // declares and assigns a value of PI.

char a = 'a'; // the char variable a is initialized with value 'a'

1. **What is variable?**

It stores the information or values.

1. **creating method with void**

Public static void methods()

Public- Access modifier

Void- return type

Methods- method name

1. **creating variable, we can create variables inside method**

**package** java1;

**public** **class** Createvar {

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

// **TODO** Auto-generated method stub

Createvar mt=**new** Createvar();

mt.methods();

}

**public** **void** methods()

{

**int** i=7;

System.***out***.println("the value of i is:"+i);

}

}

1. **creating method with return data type, we can return int/string/double/float/date etc**

**package** java1;

**public** **class** Returnvalue {

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

// **TODO** Auto-generated method stub

Returnvalue rv=**new** Returnvalue();

**int** z=rv.methodsr();

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

String s=rv.methods();

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

}

**public** **int** methodsr()

{

**int** i=2,j=8;

**int** k=i+j;

**return** k;

}

**public** String methods()

{

String h,w,l,r=**null**;

**return** r;

}

}

1. **method that will return hard coded value**
2. **method that will return property value**

Object.values(obj)

1. **creating method with void and parameter**

**public** **void** parameterr(**int** a, **int** b)

{

**int** c=a+b;

}

int a, int b - list of parameters

1. **creating static property**

Public class program2

{

Public static string name = “I am static variable”;

}

1. **creating static method**

**package** java1;

**public** **class** Staticmethod

{

// Static Variable

**static** **int** *a*,*b*,*c*,*d*;

// Static Method

**static** **void** staticMethod()

{

*a*=9;

*b*=6;

*c*=*a*+*b*;

*d*=*a*-*b*;

}

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

// accessing static variables

Staticmethod.*staticMethod*();

System.***out***.println("The addition of a&b is:"+*c*);

System.***out***.println("The subtraction of a &b is:"+*d*);

}

}

1. **creating object**

Classname obj=new Classname();

1. **calling method with void**

**package** java1;

**public** **class** Methods {

**public** **void** parameterr()

{

**int** a=9, b=2, c;

c=a+b;

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

}

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

// **TODO** Auto-generated method stub

Methods mt=**new** Methods();

mt.parameterr();

}

}

1. **calling method with no return and no parameter**

**package** java1;

**public** **class** Norenopa {

// No return and no parameters

**public** **void** Sum()

{

// inputs

**int** x=1;

**int** y=2;

// process

**int** z=x+y;

// output

System.***out***.println("Addition value: "+z);

}

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

{

// Declare the class object

Norenopa rp=**new** Norenopa();

rp.Sum();

}

1. **calling method with return and no parameter**

**package** java1;

**public** **class** Norenopa {

**public int** Sum()

{

// inputs

**int** x=1;

**int** y=2;

// process

**int** z=x+y;

// output

**return** z;

}

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

{

// Declare the class object

Norenopa rp=**new** Norenopa();

**int** t=rp.Sum();

System.***out***.println("Addition values : "+t);

}

}

1. **calling method with return and parameter**

**package** java1;

**public** **class** Norenopa {

**public** **int** Sum(**int** x,**int** y)

{

// inputs

**int** a=x;

**int** b=y;

// process

**int** c=a+b;

// output will be stored in c

**return** c;

}

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

{

// Declare the class object

Norenopa rp=**new** Norenopa();

**int** q=rp.Sum(15,1);

System.***out***.println("Addition values : "+q);

}

1. **calling method with return and storing the return data**

public int[] method()

{

int z[] = {1,2,3,5};

return z;

}

1. **calling static method**

**package** java1;

**public** **class** Staticmethod

{

// Static Variable

**static** **int** *a*,*b*,*c*,*d*;

// Static Method

**static** **void** staticMethod()

{

*a*=9;

*b*=6;

*c*=*a*+*b*;

*d*=*a*-*b*;

}

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

// accessing static variables

Staticmethod.*staticMethod*();

System.***out***.println("The addition of a&b is:"+*c*);

System.***out***.println("The subtraction of a &b is:"+*d*);

}

}

1. **create multiple classes under a package**

Right click on the package and create new class likewise you can create number of classes under one package

1. **write code to handle exceptions with try/catch/finally**

**package** java1;

**import** java.io.IOException;

**import** java.sql.SQLException;

**public** **class** Exceptions {

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

// **TODO** Auto-generated method stub

**try**

{

**int** i=1;

**if**(i==1)

**throw** **new** IOException();

**else**

**throw** **new** SQLException();

}**catch**(SQLException ex){

System.***out***.println(ex +"handled");

}**catch**(IOException ex){

System.***out***.println(ex +"handled");

}

}

}

1. **what is final keyword, create final class, final method, final property**

Final is reserved keyword in java to restrict the user and it can be applied to member variables, methods, class and local variables.

If you make a variable as final, you cannot change the value of final variable.

**package** java1;

**public** **class** Finaldemo {

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

// **TODO** Auto-generated method stub

**final** **int** i =3; //i value can’t change as it’s a

}

}

1. **write code for interface and create class to implement that interface**

**package** Interfacepractice;

**public** **interface** Clientclass {

// declaring methods without implementation

**public** **void** paycreditcard();

**public** **void** transferbalance();

**public** **void** checkingbalance();

}

**package** Interfacepractice;

**public** **class** Developing **implements** Clientclass{

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

// **TODO** Auto-generated method stub

Developing D=**new** Developing();

D.checkingbalance();

D.paycreditcard();

D.transferbalance();

D.login();

// It only calls Clientclass methods

Clientclass Cc = **new** Developing(); // Run time polymorphism

Cc.checkingbalance();

//Cc.login(); // gets error if we call a method from Developing class

}

**public** **void** paycreditcard() {

// **TODO** Auto-generated method stub

//Implementing the method

System.***out***.println("Pay credit card implemented");

}

@Override

**public** **void** transferbalance() {

// **TODO** Auto-generated method stub

//Implementing the method

System.***out***.println("transferbalance implemented");

}

@Override

**public** **void** checkingbalance() {

//Implementing the method

System.***out***.println("checkingbalance implemented");

}

**public** **void** login(){

System.***out***.println("login");

}

}

1. **implement method overloading**

**package** java1;

**public** **class** Overloading {

**void** test()

{

System.***out***.println("No parameters");

}

// Overload test for one integer parameter.

**void** test(**int** a) {

System.***out***.println("a: " + a);

}

// Overload test for two integer parameters.

**void** test(**int** a, **int** b) {

System.***out***.println("a and b: " + a + " " + b);

}

// overload test for a double parameter

**double** test(**double** a) {

System.***out***.println("double a: " + a);

**return** a\*a;

}

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

// **TODO** Auto-generated method stub

Overloading ol=**new** Overloading();

ol.test();

ol.test(6);

ol.test(3, 6);

}

}

1. **implement method overriding**

**package** java1;

**public** **class** Vehicle

{

**public** **void** over()

{

System.***out***.println("Vehicle class");

}

1. }

**package** java1;

**public** **class** Riding **extends** Vehicle {

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

// **TODO** Auto-generated method stub

Riding rd=**new** Riding();

rd.over();

}

}

1. **implementing polymorphism**

**package** java1;

**public** **class** Polymorphism {

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

// **TODO** Auto-generated method stub

Student ch=**new** Student();

ch.play();

}

}

**interface** Childrens//create interface

{

**public** **void** play();//declare method

}

**class** Student **implements** Childrens//implements interface

{

**public** **void** play()//define method

{

System.***out***.println("Students are playing football");

}

}

1. **implementing interface**

**package** Interfacepractice;

**public** **class** Developing **implements** Clientclass{

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

// **TODO** Auto-generated method stub

Developing D=**new** Developing();

D.checkingbalance();

D.paycreditcard();

D.transferbalance();

D.login();

// It only calls Clientclass methods

Clientclass Cc = **new** Developing(); // Run time polymorphism

Cc.checkingbalance();

//Cc.login(); // gets error if we call a method from Developing class

}

**public** **void** paycreditcard() {

// **TODO** Auto-generated method stub

//Implementing the method

System.***out***.println("Pay credit card implemented");

}

@Override

**public** **void** transferbalance() {

// **TODO** Auto-generated method stub

//Implementing the method

System.***out***.println("transferbalance implemented");

}

@Override

**public** **void** checkingbalance() {

//Implementing the method

System.***out***.println("checkingbalance implemented");

}

**public** **void** login(){

System.***out***.println("login");

}

}

1. **How to do inheritance in java (using extend keyword)**

**package** Inheritance;

**public** **class** ParentIn {

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

// **TODO** Auto-generated method stub

}

**int** i=38;

**public** **void** city()

{

System.***out***.println("Hyderabad");

}

**public** **void** country()

{

System.***out***.println("India");

}

}

**package** Inheritance;

**public** **class** Childinheri **extends** ParentIn {

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

// **TODO** Auto-generated method stub

// creating an object for child

Childinheri Ci= **new** Childinheri();

Ci.city(); // calling the parent class methods

Ci.country();

System.***out***.println(Ci.i); // printing the integer value in parent class

}

}

1. **write code for creating abstract class**

To create an abstract class, you use the abstract on the class declaration and include at least one abstract method.

Ex: public abstract class Ball

     {

      Public abstract int hit(int batspeed);

     }

**package** java1;

**public** **class** Abstractc {

**private** String name;

**private** **int** number;

**private** String address;

**public** Abstractc(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;

}

}

1. **write code to add items to integer, string array**

**package** java1;

**public** **class** Arrayeg {

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

// **TODO** Auto-generated method stub

**int** sum=0;

**int** c[]={2,4,5,6,7};

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

{

// adding array values using sum integer

sum=sum+c[i];

// To print the value it checked

System.***out***.println(c[i]);

// Printing the index of a value

**if**(c[i]==5)

{

**break**; // it will stop executing after getting required value

}

}

System.***out***.println("The sum of array list is:"+sum);

}

}

1. **write code to retrieve items from integer, string array**

int[] nums = {-5,1,2,11,3};

Arrays.sort(nums);

String[] a=Arrays.toString(nums).split("[\\[\\]]")[1].split(", ");

System.out.println(Arrays.toString(a));

1. **write code to add items to ArrayList collection**

**package** javaColl;

**import** java.util.ArrayList;

**public** **class** Arraylisteg {

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

// **TODO** Auto-generated method stub

// creating object for default arraylist class

ArrayList<String> al= **new** ArrayList<String>();

//adding strings to the class

al.add("sravanthi");

al.add("ujwal");

// it will print all the added value as array

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

}

}

1. **write code to retrieve items from arraylist**

**package** javaColl;

**import** java.util.ArrayList;

**public** **class** Arraylisteg {

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

// **TODO** Auto-generated method stub

// creating object for default arraylist class

ArrayList<String> al= **new** ArrayList<String>();

//adding strings to the class

al.add("sravanthi");

al.add("ujwal");

// it will print all the added value as array

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

//adding value at particular index

al.add(1, "srinivas");

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

// retrieving elements from the arraylist

System.***out***.println(al.get(2));

}

}

1. **write code to add items HashMap**

**package** javaColl;

**import** java.util.HashMap;

**public** **class** hashmap {

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

// **TODO** Auto-generated method stub

HashMap<Integer,String> hm=**new** HashMap<Integer,String>();

hm.put(0, "one");

hm.put(1, "Two");

hm.put(2, "three");

System.***out***.println(hm.get(2));

hm.remove(2);

System.***out***.println(hm.get(2));

}

}

1. **write code to retrieve items HashMap**

Iterator it = sn.iterator();

**while**(it.hasNext())

{

System.***out***.println(it.next());

Map.Entry mp= (Map.Entry)it.next();

System.***out***.println(mp.getKey());

System.***out***.println(mp.getValue());

}

1. **Write code to add items to hashset**

**package** javaColl;

**import** java.util.HashSet;

**public** **class** Hashseteg {

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

// **TODO** Auto-generated method stub

// Hashset, treeset, linkedhashset implements set interface

// does not accept duplicates values

// elements will not stored in sequential order

HashSet<String> hs=**new** HashSet<String>();

// adding values to the hashset

hs.add("java");

hs.add("c++");

hs.add("python");

printing the values

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

// it will remove c++

hs.remove("c++");

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

// checking whether the hashset is empty or not

System.***out***.println(hs.isEmpty());

// printing the size of hashset

System.***out***.println(hs.size());

}

}

1. **Write code to retrieve items to hashset**

Iterator<String> is= hs.iterator();

**while**(is.hasNext())

{

System.***out***.println(is.next());

}

1. **Difference between string, string buffer, string builder with example**

**String**is immutable (once created can’t be changed) object. The object created as a String is stored in the Constant String Pool

**String Buffer**is mutable means one can change the value of the object. The object created through String Buffer is stored in the heap

**String Builder**is same as the String Buffer , that is it stores the object in heap and it can also be modified . The main difference between the String Buffer and String Builder is thatString Builder is also not thread safe.

1. **write code to connect to JDBC to get rows from employee table**

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class Jdbcdemo {

public static void main(String[] args) throws SQLException {

// TODO Auto-generated method stub

String host="127.0.0.1";

String port="3306";

Connection con=DriverManager.getConnection("jdbc:mysql://" + host + ":"+ port + "/Qademo","root","mysql");

Statement s=con.createStatement();

ResultSet rs= s.executeQuery("select \* from Employeeinfo;");

while(rs.next())

{

System.out.println(rs.getString("name"));

System.out.println(rs.getString("location"));

System.out.println(rs.getString("id"));

System.out.println(rs.getString("age"));

}

}

}

1. **write a code to read from excel file (jexcel API)**

**package** excelDemo;

**import** java.io.File;

**import** java.io.IOException;

**import** jxl.Workbook;

**import** jxl.read.biff.BiffException;

**public** **class** ReadExcel {

**public** **static** **void** main(String[] args) **throws** BiffException, IOException {

// specifying the location of the excel sheet

File src=**new** File("/Users/sravanthigokula/Documents/TestData.xls");

System.***out***.println("Excel Located");

// load workbook method to store excel sheet

Workbook wb= Workbook.*getWorkbook*(src);

System.***out***.println("workbook loaded");

// load sheet (index starts with 0) get call(index no's

String data00= wb.getSheet(0).getCell(0,0).getContents();

String data01= wb.getSheet(0).getCell(0,1).getContents();

//printing the stored value in data00

System.***out***.println("Data is: "+data00);

System.***out***.println("Data is: "+data01);

// to count the no . of rows

**int** rows= wb.getSheet(0).getRows();

System.***out***.println("Total no of rows is: "+rows);

**int** columns= wb.getSheet(0).getColumns();

System.***out***.println("Total no of rows is: "+columns);

}

}

1. **write a code to write the data to excel file (jexcel API)**

**package** excelDemo;

**import** java.io.File;

**import** java.io.IOException;

**import** jxl.Workbook;

**import** jxl.read.biff.BiffException;

**import** jxl.write.Label;

**import** jxl.write.WritableSheet;

**import** jxl.write.WritableWorkbook;

**import** jxl.write.WriteException;

**import** jxl.write.biff.RowsExceededException;

**public** **class** Writedata {

**public** **static** **void** main(String[] args) **throws** BiffException, IOException, RowsExceededException, WriteException {

// **TODO** Auto-generated method stub

File src=**new** File("/Users/sravanthigokula/Documents/CreateExcel.xls");

System.***out***.println("Excel Located");

// load workbook method to store excel sheet

WritableWorkbook wb= Workbook.*createWorkbook*(src);

System.***out***.println("workbook created");

wb.createSheet("Write",0);

WritableSheet ws = wb.getSheet(0);

Label data1 = **new** Label(0,0,"testdata@gmail.com");

ws.addCell(data1);

Label data2 = **new** Label(0,1,"testdat@gmail.com");

ws.addCell(data2);

Label data3 = **new** Label(1,0,"testdata");

ws.addCell(data3);

Label data4 = **new** Label(1,1,"testdat");

ws.addCell(data4);

wb.write();

wb.close();

}

}

1. **how to update the data into XML file and read data from XML file**