**1 .Install Java: (day one):**

**Download Java**

The File Download dialog box appears prompting you to run or save the download file 

* To run the installer, click **Run**.
* To save the file for later installation, click **Save**.   
  Choose the folder location and save the file to your local system.   
  Tip: Save the file to a known location on your computer, for example, to your desktop.  
  Double-click on the saved file to start the installation process.
* The installation process starts. Click the **Install**button to accept the license terms and to continue with the installation.

2 .Install eclipse

Unzip eclipse-standard-luna-R-win32.zip, the file that you just downloaded and moved. ...

Create a shortcut on your desktop to the eclipse.exe file in this eclipse folder: ...

Double-click the shortcut to Eclipse that you just created above. ...

Click OK.

3. create workspace

Press the Next button and then Browse for the old projects you would like to import.

Check "Copy projects into **workspace**" to make a copy.

In Preferences->**Workspaces**, make sure that 'Prompt for **Workspace** on startup' is checked.

4. Create project

File -> Project

We call Project is program

1. create .java file/class

Project -> new class and give extension .java

Class Employee {

}

5. How to create packages and what is best way to give name

From solution explorer, select project, right click and select package

Ex: companyname.projectname.foldername (this is common naming standard)

**6)what is main method will do?**

Main method is starting point of program

**7)what is data type and different data types**

Data types represent the different values to be stored in the variable. In java, there are two types of data types:

* Primitive data types
* Non-primitive data types

Int

Double

Float

**8)creating property/data members**

int salary

**9) creating method with void**

public class ExampleVoid {

public static void main(String[] args) {

methodRankPoints(255.7);

}

public static void methodRankPoints(double points) {

if (points >= 202.5) {

System.out.println("Rank:A1");

}else if (points >= 122.4) {

System.out.println("Rank:A2");

}else {

System.out.println("Rank:A3");

}

}

}

**10)creating method with void and parameter**

public class ExampleVoid {

public static void main(String[] args) {

methodRankPoints(255.7);

}

public static void methodRankPoints(double points)

{

if (points >= 202.5) {

System.out.println("Rank:A1");

}else if (points >= 122.4) {

System.out.println("Rank:A2");

}else {

System.out.println("Rank:A3");

}

}

}

**11) creating method with return data type**

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);

}

/\*\* returns the minimum of two numbers \*/

public static int minFunction(int n1, int n2) {

int min;

if (n1 > n2)

min = n2;

else

min = n1;

return min;

}

}

12) creating method with return data type 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);

}

/\*\* returns the minimum of two numbers \*/

public static int minFunction(int n1, int n2) {

int min;

if (n1 > n2)

min = n2;

else

min = n1;

return min;

}

}

**13)creating variable**

public class Bicycle {

private int cadence;

private int gear;

private int speed;

public Bicycle(int startCadence, int startSpeed, int startGear) {

gear = startGear;

cadence = startCadence;

speed = startSpeed;

}

**14)creating static property**

**public** **class** StaticExample2

{

**static** **int** i=10;

**int** j=5;

**public** **static** **void** methodOne()

{

System.**out**.println("This is static method");

}

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

{

StaticExample2.methodOne();

System.**out**.println(StaticExample2.i);

StaticExample2 obj=**new** StaticExample2();

System.**out**.println(obj.j);

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

}

}

**15) creating static method?**

Public class StaticDemo

{

Public static void main(String args[])

{

Abc.show();//calling static method

}

}

Class Abc

{

Public static void show() //static method

{

System.out.println(“Hi”)

}

}

https://www.youtube.com/watch?v=-TxHjyC8WY4

**16) creating object?**

Classname obj=new Classname();

Ex: Animal b1=new Animal();

**17) calling method with no return**

public class Program {

static void displayPassword(String password) {

// Write the password to the console.

System.out.println("Password: " + password);

// Return if our password is long enough.

if (password.length() >= 5) {

**return**;

}

System.out.println("Password too short!");

// An implicit return is here.

}

public static void main(String[] args) {

displayPassword("furball");

displayPassword("cat");

}

}

**18) calling method with no return and parameter**

**amoolya**

**public class ExampleVoid {**

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

**methodRankPoints(255.7);**

**}**

**public static void methodRankPoints(double points) {**

**if (points >= 202.5) {**

**System.out.println("Rank:A1");**

**}else if (points >= 122.4) {**

**System.out.println("Rank:A2");**

**}else {**

**System.out.println("Rank:A3");**

**}**

**}**

**}**

**19)calling method with return and no parameter**

**Package instanceofjava;**

**class sample{**

**public int add(){**

**int a=40;**

**int b=50;**

**int c=a+b;**

**return c;**

**}**

**public static void main(String args[]) // ->method prototype.**

**{**

**sample obj= new sample();**

**int x=obj.add();**

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

**}**

**}**

**20)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);**

**}**

**/\*\* returns the minimum of two numbers \*/**

**public static int minFunction(int n1, int n2) {**

**int min;**

**if (n1 > n2)**

**min = n2;**

**else**

**min = n1;**

**return min;**

**}**

**}**

**21)calling method with return and storing the return data**

import java.io.\*;  
import BreezyGUI.\*;  
  
public class method1test  
{  
     public static void main(String[ ] args)  
    {  
          greeting(5);  //first method call  
          int response;  
          response = getNumber();   //second method call  
          greeting (response);          //first method call again  
    }

     //First Method for greeting  
     public static void greeting(int x)  
    {  
         int i;  
         for(i = 0; i < x; i++)  
         {  
               System.out.print("Hi ");  
         }  
         System.out.println( );  
    }

    //Second method which returns a number  
    public static int getNumber( )  
    {  
         int number;  
        do  
        {  
            number = Console.readInt("\nPlease enter value(1-10): ");  
         }  
        while ((number < 1) || (number > 10));  //error trap  
        return number;  //return the number to main  
    }  
}

**22)calling static method**

**Public class StaticDemo**

**{**

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

**{**

**Abc.show();//call**ing static method

}

}

Class Abc

{

Public static void show() //static method

{

System.out.println(“Hi”)

}

}

**23)using static property**

**package** staticExample;

**public** **class** StaticExampleTwo

{

**static** **int** i=10;

**int** j=5;

**public** **static** **void** methodOne()

{

System.**out**.println("This is static method");

}

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

{

StaticExampleTwo.methodOne();

//static method can access static variable

System.**out**.println(StaticExampleTwo.i);

StaticExampleTwo obj=**new** StaticExampleTwo();

// a static method can access instance variable through object reference

System.**out**.println(obj.j);

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

}

}

**24 create classes under multiple packages**

**25) calling classes under different packages**

package animals;

interface Animal {

public void eat();

public void travel();

}

package animals;

/\* File name : MammalInt.java \*/

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();

**}**

**}**

**import animals.\*; OR be more specific import animals.MammalInt;**

**26)write code to handle exceptions with try/catch/finally**

**A.Code for try and Catch**

**import java.util.\*;**

**import java.io.\*;**

**public class Excep\_ex5**

**{**

**static Scanner in=new Scanner(System.in);**

**static PrintStream out=new PrintStream(System.out);**

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

**{**

**int x, y, res;**

**try**

**{**

**out.println("enter the x");**

**x=in.nextInt();**

**out.println("enter the y");**

**y=in.nextInt();**

**res=x/y;**

**out.println("Quotient :" + res);**

**}**

**catch(Exception e)**

**{**

**if( e instanceof ArithmeticException)**

**out.println("Exception :" + e.toString() );**

**else if( e instanceof InputMismatchException )**

**out.println("Exception :" + e.toString() );**

**}**

**}**

**}**

**27)Code for Catch and Finally keyword:**

**package javaBasics;**

**import java.io.PrintStream;**

**public class Excep\_4**

**{**

**static PrintStream out=new PrintStream(System.out);**

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

**{**

**int a, b, res=0;**

**a=10;**

**b=2;**

**try {**

**res=a/b;**

**}catch(Exception e) {**

**System.err.println(e.getMessage() );**

**}**

**finally**

**{**

**out.println("From Finally Block");**

**out.println("Result :"+res);**

**}**

**}**

**}**

**28)what is final keyword**

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 can not change the value of final variable.

http://www.javatpoint.com/final-keyword

**Q)write code for interface and create class to implement that interface**

**interface**

interface MyInterface

{

public void method1();

public void method2();

}

class XYZ implements MyInterface

{

public void method1()

{

System.out.println("implementation of method1");

}

public void method2()

{

System.out.println("implementation of method2");

}

public static void main(String arg[])

{

MyInterface obj = new XYZ();

obj. method1();

}

}

**29)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);

}

http://www.dummies.com/programming/java/create-an-abstract-class-in-java/  
**30)implement method overloading**

**A.Code for method overloading:**

**package javaBasics;**

**/// example on method over-loading**

**import static java.lang.System.out;**

**class Box**

**{**

**double l, b, h;**

**public void setData(double x, double y, double z) {**

**l=x;**

**b=y;**

**h=z;**

**}**

**public void setData(double x) {**

**l=b=h=x;**

**}**

**public void setData() {**

**l=b=h=1;**

**}**

**public double volume() {**

**return(l\*b\*h);**

**}**

**}**

**public class Testclass\_9**

**{**

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

**{**

**Box b=new Box();**

**b.setData(1,2,3);**

**out.println("Volume :"+ b.volume());**

**Box bb=new Box();**

**bb.setData(3);**

**out.println("Volume :"+ bb.volume());**

**}**

**}**

**31)implement method overriding**

class Human{

public void eat()

{

System.out.println("Human is eating");

}

}

class Boy extends Human{

public void eat(){

System.out.println("Boy is eating");

}

public static void main( String args[]) {

Boy obj = new Boy();

obj.eat();

}

}

**32) implementing polymorphism**

**class** Bike{

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

}

**class** Splender **extends** Bike{

**void** run(){System.out.println("running safely with 60km");}

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

    Bike b = **new** Splender();//upcasting

    b.run();

  }

}

**33) implementing interface**

interface MyInterface

{

public void method1();

public void method2();

}

class XYZ implements MyInterface

{

public void method1()

{

System.out.println("implementation of method1");

}

public void method2()

{

System.out.println("implementation of method2");

}

public static void main(String arg[])

{

MyInterface obj = new XYZ();

obj. method1();

}

}

**34) Write a code to save data into excel file and read from excel file (POI and jexcel API)**

**package** writer;

**import** java.io.File;

**import** java.io.IOException;

**import** java.util.Locale;

**import** jxl.CellView;

**import** jxl.Workbook;

**import** jxl.WorkbookSettings;

**import** jxl.format.UnderlineStyle;

**import** jxl.write.Formula;

**import** jxl.write.Label;

**import** jxl.write.Number;

**import** jxl.write.WritableCellFormat;

**import** jxl.write.WritableFont;

**import** jxl.write.WritableSheet;

**import** jxl.write.WritableWorkbook;

**import** jxl.write.WriteException;

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

**public** **class** **WriteExcel** {

**private** WritableCellFormat timesBoldUnderline;

**private** WritableCellFormat times;

**private** String inputFile;

**public** **void** setOutputFile(String inputFile) {

this.inputFile = inputFile;

}

**public** **void** write() **throws** IOException, WriteException {

File file = **new** File(inputFile);

WorkbookSettings wbSettings = **new** WorkbookSettings();

wbSettings.setLocale(**new** Locale("en", "EN"));

WritableWorkbook workbook = Workbook.createWorkbook(file, wbSettings);

workbook.createSheet("Report", 0);

WritableSheet excelSheet = workbook.getSheet(0);

createLabel(excelSheet);

createContent(excelSheet);

workbook.write();

workbook.close();

}

**private** **void** createLabel(WritableSheet sheet)

**throws** WriteException {

// Lets create a times font

WritableFont times10pt = **new** WritableFont(WritableFont.TIMES, 10);

// Define the cell format

times = **new** WritableCellFormat(times10pt);

// Lets automatically wrap the cells

times.setWrap(true);

// create create a bold font with unterlines

WritableFont times10ptBoldUnderline = **new** WritableFont(

WritableFont.TIMES, 10, WritableFont.BOLD, false,

UnderlineStyle.SINGLE);

timesBoldUnderline = **new** WritableCellFormat(times10ptBoldUnderline);

// Lets automatically wrap the cells

timesBoldUnderline.setWrap(true);

CellView cv = **new** CellView();

cv.setFormat(times);

cv.setFormat(timesBoldUnderline);

cv.setAutosize(true);

// Write a few headers

addCaption(sheet, 0, 0, "Header 1");

addCaption(sheet, 1, 0, "This is another header");

}

**private** **void** createContent(WritableSheet sheet) **throws** WriteException,

RowsExceededException {

// Write a few number

**for** (**int** i = 1; i < 10; i++) {

// First column

addNumber(sheet, 0, i, i + 10);

// Second column

addNumber(sheet, 1, i, i \* i);

}

// Lets calculate the sum of it

StringBuffer buf = **new** StringBuffer();

buf.append("SUM(A2:A10)");

Formula f = **new** Formula(0, 10, buf.toString());

sheet.addCell(f);

buf = **new** StringBuffer();

buf.append("SUM(B2:B10)");

f = **new** Formula(1, 10, buf.toString());

sheet.addCell(f);

// now a bit of text

**for** (**int** i = 12; i < 20; i++) {

// First column

addLabel(sheet, 0, i, "Boring text " + i);

// Second column

addLabel(sheet, 1, i, "Another text");

}

}

**private** **void** addCaption(WritableSheet sheet, **int** column, **int** row, String s)

**throws** RowsExceededException, WriteException {

Label label;

label = **new** Label(column, row, s, timesBoldUnderline);

sheet.addCell(label);

}

**private** **void** addNumber(WritableSheet sheet, **int** column, **int** row,

Integer integer) **throws** WriteException, RowsExceededException {

Number number;

number = **new** Number(column, row, integer, times);

sheet.addCell(number);

}

**private** **void** addLabel(WritableSheet sheet, **int** column, **int** row, String s)

**throws** WriteException, RowsExceededException {

Label label;

label = **new** Label(column, row, s, times);

sheet.addCell(label);

}

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

WriteExcel test = **new** WriteExcel();

test.setOutputFile("c:/temp/lars.xls");

test.write();

System.out

.println("Please check the result file under c:/temp/lars.xls ");

}

}

Read the excel

**package** reader;

**import** java.io.File;

**import** java.io.IOException;

**import** jxl.Cell;

**import** jxl.CellType;

**import** jxl.Sheet;

**import** jxl.Workbook;

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

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

**private** String inputFile;

**public** **void** setInputFile(String inputFile) {

this.inputFile = inputFile;

}

**public** **void** read() **throws** IOException {

File inputWorkbook = **new** File(inputFile);

Workbook w;

**try** {

w = Workbook.getWorkbook(inputWorkbook);

// Get the first sheet

Sheet sheet = w.getSheet(0);

// Loop over first 10 column and lines

**for** (**int** j = 0; j < sheet.getColumns(); j++) {

**for** (**int** i = 0; i < sheet.getRows(); i++) {

Cell cell = sheet.getCell(j, i);

CellType type = cell.getType();

**if** (type == CellType.LABEL) {

System.out.println("I got a label "

+ cell.getContents());

}

**if** (type == CellType.NUMBER) {

System.out.println("I got a number "

+ cell.getContents());

}

}

}

} **catch** (BiffException e) {

e.printStackTrace();

}

}

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

ReadExcel test = **new** ReadExcel();

test.setInputFile("c:/temp/lars.xls");

test.read();

}

}

**35) how to update the data into XML file and read data from XML file**

setNodeValue — takes an element (or tag) name, a String value, and the list of nodes within which the existing node can be found

addNode — takes an element (or tag) name, a String value, and the parent node for which the new node is to be a child of

protected void setNodeValue(String tagName, String value, NodeList nodes) {

    Node node = getNode(tagName, nodes);

    if ( node == null )

        return;

    // Locate the child text node and change its value

    NodeList childNodes = node.getChildNodes();

    for (int y = 0; y < childNodes.getLength(); y++ ) {

        Node data = childNodes.item(y);

        if ( data.getNodeType() == Node.TEXT\_NODE ) {

            data.setNodeValue(value);

            return;

        }

    }

}

protected void addNode(String tagName, String value, Node parent) {

    Document dom = parent.getOwnerDocument();

    // Create a new Node with the given tag name

    Node node = dom.createElement(tagName);

    // Add the node value as a child text node

    Text nodeVal = dom.createTextNode(value);

    Node c = node.appendChild(nodeVal);

    // Add the new node structure to the parent node

    parent.appendChild(node);

}

**36) write code to add items to integer, string array**

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

**int[] series = {4,2};**

**series = addElement(series, 3);**

**series = addElement(series, 1);**

**}**

**static int[] addElement(int[] a, int e) {**

**a = Arrays.copyOf(a, a.length + 1);**

**a[a.length - 1] = e;**

**return a;**

**}**

**37)write code to retrieve items from integer, string array**

**import** java.lang.reflect.Array;

**import** java.util.Arrays;

**public** **class** Main {

**public** **static** **void** main(String[] argv) **throws** Exception {

**int**[] array = { 1, 2, 3 };

// Get the value of the third element.

Object o = Array.get(array, 2);

System.out.println("o:"+o);

}

}

**38)write code to add items to ArrayList collection**

package beginnersbook.com;

import java.util.ArrayList;

public class GetMethodExample {

public static void main(String[] args) {

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

al.add("pen");

al.add("pencil");

al.add("ink");

al.add("notebook");

al.add("book");

al.add("books");

al.add("paper");

al.add("white board");

System.out.println("First element of the ArrayList: "+al.get(0));

System.out.println("Third element of the ArrayList: "+al.get(2));

System.out.println("Sixth element of the ArrayList: "+al.get(5));

System.out.println("Fourth element of the ArrayList: "+al.get(3));

}

}

**39)write code to retrieve items from arraylist**

package beginnersbook.com;

import java.util.ArrayList;

public class GetMethodExample {

public static void main(String[] args) {

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

al.add("pen");

al.add("pencil");

al.add("ink");

al.add("notebook");

al.add("book");

al.add("books");

al.add("paper");

al.add("white board");

System.out.println("First element of the ArrayList: "+al.get(0));

System.out.println("Third element of the ArrayList: "+al.get(2));

System.out.println("Sixth element of the ArrayList: "+al.get(5));

System.out.println("Fourth element of the ArrayList: "+al.get(3));

}

}

**40)write code to add items HashMap**

amoolya

public void processHashMap() {

//add keys->value pairs to a hashmap:

HashMap hm = new HashMap();

hm.put(1, "godric gryfindor");

hm.put(2, "helga hufflepuff");

hm.put(3, "rowena ravenclaw");

hm.put(4, "salazaar slytherin");

//Then get data back out of it:

LinkedList ll = new LinkedList();

Iterator itr = hm.keySet().iterator();

String key = itr.next();

ll.add(key);

}

System.out.print(ll); //The key list will be printed.

}

**41)write code to retrieve items HashMap**

public void processHashMap() {

//add keys->value pairs to a hashmap:

HashMap hm = new HashMap();

hm.put(1, "godric gryfindor");

hm.put(2, "helga hufflepuff");

hm.put(3, "rowena ravenclaw");

hm.put(4, "salazaar slytherin");

//Then get data back out of it:

LinkedList ll = new LinkedList();

Iterator itr = hm.keySet().iterator();

while(itr.hasNext()) {

String key = itr.next();

ll.add(key);

}

System.out.print(ll); //The key list will be printed.

}

42)write code to connect to JDBC to get rows from employee table

package com.mkyong.jdbc;

import java.sql.DriverManager;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class JDBCStatementSelectExample {

private static final String DB\_DRIVER = "oracle.jdbc.driver.OracleDriver";

private static final String DB\_CONNECTION = "jdbc:oracle:thin:@localhost:1521:MKYONG";

private static final String DB\_USER = "user";

private static final String DB\_PASSWORD = "password";

public static void main9String[] args)

{

Try

{

} catch (SQLException e) {

System.out.println(e.getMessage());

}

}

private static void selectRecordsFromDbUserTable() throws SQLException {

Connection dbConnection = null;

Statement statement = null;

String selectTableSQL = "SELECT USER\_ID, USERNAME from DBUSER";

try {

dbConnection = getDBConnection();

statement = dbConnection.createStatement();

System.out.println(selectTableSQL);

// execute select SQL stetement

ResultSet rs = statement.executeQuery(selectTableSQL);

while (rs.next()) {

String userid = rs.getString("USER\_ID");

String username = rs.getString("USERNAME");

System.out.println("userid : " + userid);

System.out.println("username : " + username);

}

} catch (SQLException e) {

System.out.println(e.getMessage());

} finally {

if (statement != null) {

statement.close();

}

if (dbConnection != null) {

dbConnection.close();

}

}

}

private static Connection getDBConnection() {

Connection dbConnection = null;

try {

Class.forName(DB\_DRIVER);

} catch (ClassNotFoundException e) {

System.out.println(e.getMessage());

}

try {

dbConnection = DriverManager.getConnection(DB\_CONNECTION, DB\_USER,DB\_PASSWORD);

return dbConnection;

}

Catch(SQLException e)

System.out.println(e.getMessage());

}

return dbConnection;

}

}

**43)create Employee class**

**44)Add employee class to list collection**

45**)create method that return list of employee collection**

import java.util.ArrayList;

public class MainClass {

public static void main(String[] a) {

ArrayList<Employee> emps = new ArrayList<Employee>();

emps.add(new Employee("A", "G"));

emps.add(new Employee("T", "A"));

emps.add(new Employee("K", "J"));

System.out.println(emps);

Employee e = emps.get(1);

e.setLastName("new");

System.out.println(emps);

}

}

class Address {

}

class Employee {

private String lastName;

private String firstName;

private Double salary;

public Address address;

public Employee(String lastName, String firstName) {

this.lastName = lastName;

this.firstName = firstName;

this.address = new Address();

}

public String getLastName() {

return this.lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public String getFirstName() {

return this.firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public double getSalary() {

return this.salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

}

47.What is a variable?

A variable provides us with named storage that our programs can manipulate. Each variable in Java has a specific type, which determines the size and layout of the variable's memory; the range of values that can be stored within that memory; and the set of operations that can be applied to the variable.

48. method that will return property value

Obj.values(obj);