1) Steps to install java

The procedure to install Java broadly consists of:

[Download and Install](https://java.com/en/download/help/windows_manual_download.xml#download)

[Test Installation](https://java.com/en/download/help/windows_manual_download.xml#test)

» [Windows System Requirements](https://java.com/en/download/help/sysreq.xml)   
  
Download and Install

It is recommended; before you proceed with online installation you may want to disable your Internet firewall. In some cases the default firewall settings are set to reject all automatic or online installations such as the Java online installation Go to the [Manual download](https://java.com/en/download/manual.jsp) page

Click on Windows Online

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.

Oracle has partnered with companies that offer various products. The installer may present you with option to install these programs when you install Java. After ensuring that the desired programs are selected, click the Next button to continue the installation.

A few brief dialogs confirm the last steps of the installation process; click Close on the last dialog. This will complete Java installation process.

Info iconDetect older versions (8u20 and later versions). Starting with Java 8 Update 20 (8u20), on Windows systems, the Java Uninstall Tool is integrated with the installer to provide an option to remove older versions of Java from the system. The change is applicable to 32 bit and 64 bit Windows platforms.

Notifications about disabled Java and restoring prompts

The installer notifies you if Java content is disabled in web browsers, and provides instructions for enabling it. If you previously chose to hide some of the security prompts for applets and Java Web Start applications, the installer provides an option for restoring the prompts. The installer may ask you to reboot your computer if you chose not to restart an internet browser when it prompted you to do so.

Test Installation

To test that Java is installed and working properly on your computer, run this [test applet](https://java.com/en/download/help/testvm.xml).

NOTE: You may need to restart (close and re-open) your browser to enable the Java installation in your browser.

2) Steps to install eclipse.

Downloading Eclipse

You can download eclipse from <http://www.eclipse.org/downloads/>. The download page lists a number of flavors of eclipse.



The capabilities of each packaging of eclipse are different. Java developers typically use Eclipse Classic or Eclipse IDE for developing Java applications.

The drop down box in the right corner of the download page allows you to set the operating system on which eclipse is to be installed. You can choose between Windows, Linux and Mac. Eclipse is packaged as a zip file.

Installing Eclipse

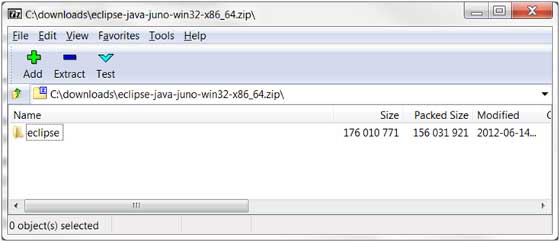
To install on windows, you need a tool that can extract the contents of a zip file. For example you can use −

[7-zip](http://www.7-zip.org/)

[PeaZip](http://sourceforge.net/projects/peazip/)

[IZArc](http://www.izarc.org/)

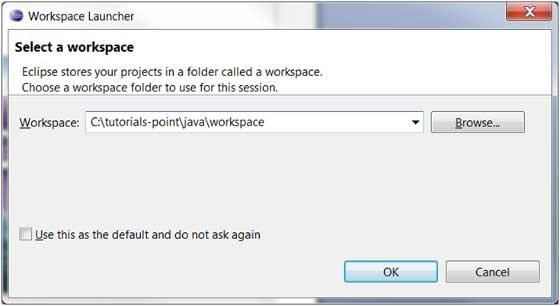
Using any one of these tools, extract the contents of the eclipse zip file to any folder of your choice.



Launching Eclipse

On the windows platform, if you extracted the contents of the zip file to c:\, then you can start eclipse by using c:\eclipse\eclipse.exe

When eclipse starts up for the first time it prompts you for the location of the workspace folder. All your data will be stored in the workspace folder. You can accept the default or choose a new location.



If you select "Use this as the default and do not ask again", this dialog box will not come up again. You can change this preference using the Workspaces Preference Page.

3) Steps to create workspace

about Eclipse Workspace

The eclipse workspace contains resources such as −

Projects

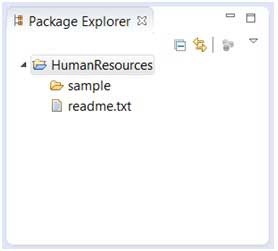
Files

Folders

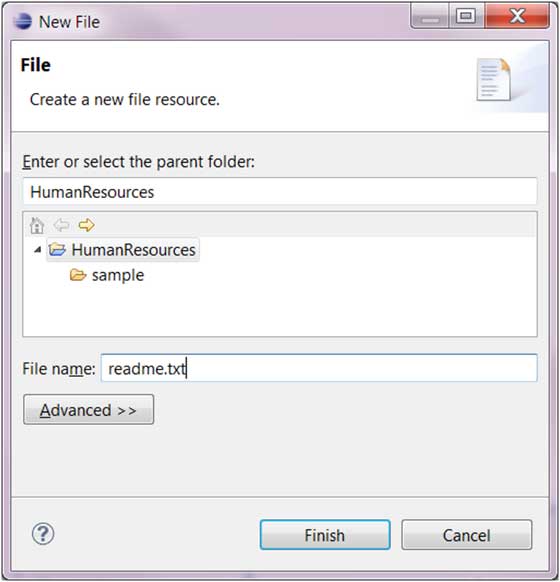
The workspace has a hierarchical structure. Projects are at the top level of the hierarchy and inside them you can have files and folders. Plug-ins use an API provided by the resources plug-in to manage the resources in the workspace.

UI Elements for Managing the Workspace

Users use the functionality provided by views, editors and wizard to create and manage resources in the workspace. One among the many views that show the content of the workspace is the Project Explorer view.



The File Wizard (File → New → File) can be used to create a new file.



The Folder Wizard (File → New → Folder) can be used to create a new folder.

4) Steps to create project

Opening the New Java Project wizard

The New Java Project wizard can be used to create a new java project. There are many ways to open this wizard −

By clicking on the File menu and choosing New →Java Project.

By right clicking anywhere in the Project Explorer and selecting New → Java Project.

By clicking on the New button (New Button ) in the Tool bar and selecting Java Project.

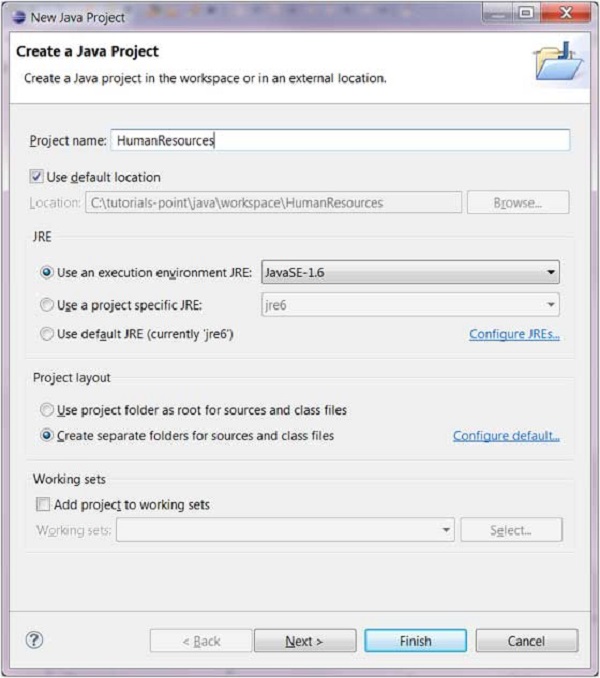
Using the New Java Project wizard

The New Java Project Wizard has two pages. On the first page −

Enter the Project Name

Select the Java Runtime Environment (JRE) or leave it at the default

Select the Project Layout which determines whether there would be a separate folder for the source codes and class files. The recommended option is to create separate folders for sources and class files.

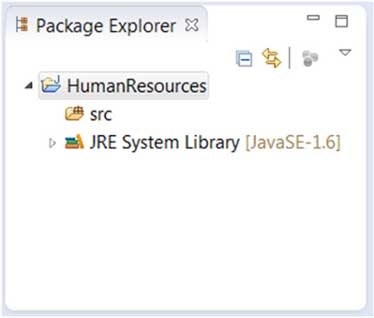


You can click on the Finish button to create the project or click on the Next button to change the java build settings.

On the second page you can change the [Java Build Settings](https://www.tutorialspoint.com/eclipse/eclipse_java_build_path.htm) like setting the Project dependency (if there are multiple projects) and adding additional jar files to the build path.

Viewing the Newly Created Project

The package explorer shows the newly created Java project. The icon that represents a Project is decorated with a J to show that it is a Java Project. The folder icon is decorated to show that it is a java source folder.



5) create .java file/class and how to create packages and what is best way to give name

Opening the New Java Class Wizard

You can use the New Java Class wizard to create a Java class. The Java Class wizard can be invoked in different ways −

By clicking on the File menu and selecting New → Class.

By right clicking in the package explorer and selecting New → Class.

By clicking on the class drop down button (Drop Down Button ) and selecting class ( Class Button).

Before bringing up the New Java Class wizard, if possible, select the package in which the class is to be created so that the wizard can automatically fill in the package name for you.

Using the New Java Class Wizard

Once the java class wizard comes up −

Ensure the source folder and package are correct.

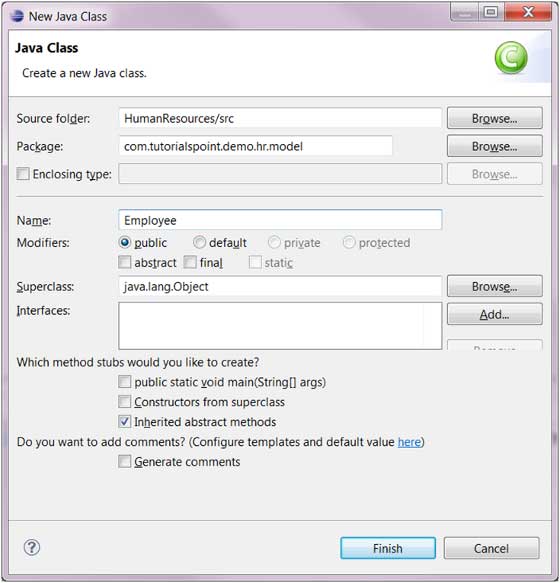
Enter the class name.

Select the appropriate class modifier.

Enter the super class name or click on the Browse button to search for an existing class.

Click on the Add button to select the interfaces implemented by this class.

Examine and modify the check boxes related to method stubs and comments.



Click the Finish button.

6) what is main method will do?

/\*\*

\* The HelloWorldApp class implements an application that

\* simply displays "Hello World!" to the standard output.

\*/

class HelloWorldApp {

public static void main(String[] args) {

System.out.println("Hello World!"); //Display the string.

}

}

Every Java application must contain a main method whose signature looks like this:

public static void main(String[] args)

The method signature for the main method contains three modifiers:

public indicates that the main method can be called by any object. [Controlling Access to Members of a Class](http://journals.ecs.soton.ac.uk/java/tutorial/java/javaOO/accesscontrol.html) covers the ins and outs of the access modifiers supported by the Java language.

static indicates that the main method is a class method. [Instance and Class Members](http://journals.ecs.soton.ac.uk/java/tutorial/java/javaOO/classvars.html)  talks about class methods and variables.

void indicates that the main method doesn't return any value.

7) creating property/data members :

The Public and Private Members of a Class

To create a field that can be accessed by only members of the same class, precede its data type with the private keyword. Here are examples:

public class House{

String propertyNumber;

private String kitchenCharacteristics;

byte Stories;

int bedrooms;

private boolean bathroomIsDirty;

double Value;

}

To create a field that can be accessed by members of the same class and members of other classes, precede its data type with the public keyword. Here are examples:

public class House{

public String propertyNumber;

private String kitchenCharacteristics;

byte Stories;

public int bedrooms;

private boolean bathroomIsDirty;

double Value;

}

8) what is data type and different data types`

Data Types in Java

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| datatype in java   |  |  |  | | --- | --- | --- | | Data Type | Default Value | Default size | | boolean | False | 1 bit | | char | '\u0000' | 2 byte | | byte | 0 | 1 byte | | short | 0 | 2 byte | | int | 0 | 4 byte | | long | 0L | 8 byte | | float | 0.0f | 4 byte | | double | 0.0d | 8 byte | |

9) What is variable?

Variable is a name of memory location. There are three types of variables in java: local, instance and static.

Types of Variable

There are three types of variables in java:

1) Local Variable

A variable which is declared inside the method is called local variable.

2) Instance Variable

A variable which is declared inside the class but outside the method, is called instance variable . It is not declared as static.

3) Static variable

A variable that is declared as static is called static variable. It cannot be local.

EX: class A{

int data=50;//instance variable

static int m=100;//static variable

void method(){

int n=90;//local variable

}

}//end of class

10) creating method with void ?

Public static int methodname()

Public static - access modifier

int - return type

11) creating variable, we can create variables inside method

Here, age is a local variable. This is defined inside pupAge() method and its scope is limited to only this method.

public class Test {

public void pupAge() {

int age = 0;

age = age + 7;

System.out.println("Puppy age is : " + age);

}

public static void main(String args[]) {

Test test = new Test();

test.pupAge();

}

}

Output:

Puppy age is :7

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

public class OddEvenSum {

public static void main(String[] args) {

int lowerbound = 1, upperbound = 1000;

int sumOdd = 0; // For accumulating odd numbers, init to 0

int sumEven = 0; // For accumulating even numbers, init to 0

int number = lowerbound;

while (number <= upperbound) {

if (number % 2 == 0) { // Even

sumEven += number; // Same as sumEven = sumEven + number

} else { // Odd

sumOdd += number; // Same as sumOdd = sumOdd + number

}

++number; // Next number

}

// Print the result

System.out.println("The sum of odd numbers from " + lowerbound + " to " + upperbound + " is " + sumOdd);

System.out.println("The sum of even numbers from " + lowerbound + " to " + upperbound + " is " + sumEven);

System.out.println("The difference between the two sums is " + (sumOdd - sumEven));

}

}

o/p

The sum of odd numbers from 1 to 1000 is 250000

The sum of even numbers from 1 to 1000 is 250500

The difference between the two sums is -500

12) method that will return hard coded value

import java.util.Scanner;

public class CommandLine {

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

System.out.print("Enter first number: ");

int firstNumber = s.nextInt();

System.out.print("Enter second number: ");

int secondNumber = s.nextInt();

int sum = firstNumber + secondNumber;

System.out.println("The result of addition was " + sum);

}

}

13) method that will return property value

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.OutputStream;

import java.util.Properties;

public class App {

public static void main(String[] args) {

Properties prop = new Properties();

OutputStream output = null;

try {

output = new FileOutputStream("config.properties");

// set the properties value

prop.setProperty("database", "localhost");

prop.setProperty("dbuser", "mkyong");

prop.setProperty("dbpassword", "password");

// save properties to project root folder

prop.store(output, null);

} catch (IOException io) {

io.printStackTrace();

} finally {

if (output != null) {

try {

output.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

}

14) creating method with return data type and parameter

package pkg;

public class ParameterizedConstructor {

int id;

String name; //declaring variables

ParameterizedConstructor(int i,String n){ //parameterized constructor

id = i;

name = n; //assigning values

}

void display()

{

System.out.println(id+" "+name);

}

public static void main(String args[]){

ParameterizedConstructor s1 = new ParameterizedConstructor(111,"Ravali");

ParameterizedConstructor s2 = new ParameterizedConstructor(222,"Ball");

s1.display(); //calling objects

s2.display();

}

}

o/p: 111 Ravali

222 Ball

15) Creating static property.

class Student{

   int rollno;

   String name;

   static String college ="HITS";  //initializing variables

   Student(int r,String n){

   rollno = r;

   name = n;

   }

 void display (){System.out.println(rollno+" "+name+" "+college);}  //printing

public static void main(String args[]){

 Student s1 = new Student(111,"Ravali");  //creating object

 Student s2 = new Student(222,"Ball");  //creating object

 s1.display();  //calling object

 s2.display(); //calling object

}

}

Output:111 Ravali HITS

222 Ball HITS

16) creating static method?

17)calling method with void

class Student1{

     int rollno;

     String name;

     static String college = "HITS";

     static void change(){  // creating static method

     college = "KITS";

     }

    Student1(int r, String n){ //creating constructor

     rollno = r;

    name = n;

  }

  //creating method name display

    void display (){

System.out.println(rollno+" "+name+" "+college);}

   public static void main(String args[]){

    Student1.change();

  //creating objects

    Student1 s1 = new Student1 (111,"Ravali");

   Student1 s2 = new Student1 (222,"Ball");

   Student1 s3 = new Student1 (333,"Prasanthi");

   s1.display();  //calling method

   s2.display();

    s3.display();

    }

}

O/P: 111 Ravali KITS

222 Ball KITS

333 Prasanthi KITS

18) creating object

Classname obj=new Classname();

Ex: Apple1 a=new Apple1();

19) Calling method with no return and parameter

class Apple1{

Apple1()//creating constructor with no parameters

{

System.out.println("Apple is created");}

public static void main(String args[]){

Apple1 a=new Apple1();  //creation of object

}

}

o/p Apple is created

20) calling method with return and parameter

class Student4{

   int id;

   String name;

         Student4(int i,String n)//parameterized constructor

{

    id = i;

    name = n;  //assigning values

    }

    void display(){

System.out.println(id+" "+name);}

public static void main(String args[]){

  Student4 s1 = new Student4(111,"Ravali");  //creation of objects and paasing values

  Student4 s2 = new Student4(222,"Ball");

   s1.display();  //calling method

   s2.display();

  }

}

o/p

111 Ravali

222 Ball

20) Calling method with return and storing the return data

|  |
| --- |
|  |
|  |  |
|  | Package callingmethodwithreturnandstoringreturndata;  public class MethodWithReturn { // always create a class name with capital letters |
|  | String printString() |
|  | { |
|  | return "returning Hard coded value"; |
|  | } |
|  |  |
|  | int printInt() |
|  | { |
|  | int k = 10; //Variable scope is within method only |
|  | return k; // returning through value |
|  | } |
|  |  |
|  | double printDouble(double d) // return type with parameter |
|  | { |
|  | return d; |
|  | } |
|  | float printfloat() |
|  | { |
|  | float k = 5.2f; // a float value should always end with f |
|  | return k; |
|  | } |
|  | public static void main(String[] args) { |
|  | MethodWithReturn m = new MethodWithReturn(); // creating an object to access the methods |
|  | String val = m.printString(); // as printString method returns a value, Created val to store the returned value |
|  | System.out.println(val); |
|  | System.out.println(m.printInt()); // can be printed directly |
|  | double d = 2.9773; |
|  | System.out.println(m.printDouble(d)); |
|  |  |
|  |  |
|  | } |
|  | } |

O/P:  returning Hard coded value

10

2.9773

 21) calling static method

Package callingstaticmethod;

|  |
| --- |
| public class StaticMethod { |
|  | static void Printit(){ |
|  | System.out.println("this is a static method"); |
|  | } |
|  | public static void main(String[] args) { |
|  | Printit(); // object is not created. static methods can be accessed directly |
|  | } |
|  |  |
|  | } |

O/P: this is a static method

22)using static property: it will maintain

|  |
| --- |
| Package usingstaticproperty; |
|  |  |
|  | public class StaticProperty { |
|  |  |
|  | static String Name="this is a static property"; |
|  | public static void main(String[] args) { |
|  | System.out.println(Name); // no need to create an object as the property is static |
|  | } |
|  |  |
|  |  |
|  | } |

O/P: this is a static property

23) create classes under multiple packages

Goto package-right click create class

In this way we can create multiple classes.

24) calling classes under different packages

25) write code to handle exceptions with try/catch/finally?

package pkg;

public class Exceptionhandling {

void test(){

try {

int x=0,y=8;

int num=y/x;

System.out.println(num);

}catch(Exception e){

System.out.println("Exception Occoured " +e);

}

finally{

System.out.println("This is finally block ");

}

}

public static void main(String[] args) {

Exceptionhandling eh=new Exceptionhandling();

eh.test();

}

}

O/P: Exception Occoured java.lang.ArithmeticException: / by zero

This is finally block

26) what is final keyword, create final class, final method, final property

If you make any variable as final, you cannot change the value of final variable(It will be constant).

1) Final Variable:

There is a final variable speedlimit, we are going to change the value of this variable, but It can't be changed because final variable once assigned a value can never be changed.

class Bike9{

 final int speedlimit=90;//final variable

 void run(){

  speedlimit=400;

 }

 public static void main(String args[]){

 Bike9 obj=new  Bike9();

 obj.run();

 }

}

o/p :compiler error because once we declare we can’t change.

2) Java final method

If you make any method as final, you cannot override it.

Example of final method

class Bike{

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

}

class Honda extends Bike{

   void run(){System.out.println("running safely with 100kmph");}

   public static void main(String args[]){

  Honda honda= new Honda();

  honda.run();

   }

}

3) Java final class

If you make any class as final, you cannot extend it.

Example of final class

final class Bike{}

class Honda1 extends Bike{

 void run(){System.out.println("running safely with 100kmph");}

  public static void main(String args[]){

 Honda1 honda= new Honda();

  honda.run();

  }

}

27) write code for interface and create class to implement that interface

28) write code for creating abstract class

29) implement method overloading?

package methodoverloading;

public class MethodOverloading {

void sum(int a,int b)

{

System.out.println(a+b);

}

void sum(int a,int b,int c)

{

System.out.println(a+b+c);

}

public static void main(String args[]){

MethodOverloading obj=new MethodOverloading();

obj.sum(10,10,10);

obj.sum(20,20);

}

}

o/p: 30

40

30) implement method overriding

**package** com.Overriding;

**class** Vehicle{

**void** run(){

System.***out***.println("Vehicle is running");}

}

**class** Bike2 **extends** Vehicle{ // method overriding

**void** run()

{

System.***out***.println("Bike is running safely");

}

}

**public** **class** MethodOverriding {

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

Bike2 r = **new** Bike2();

Vehicle b = **new** Vehicle();

r.run();

b.run();

}

}

o/p

Bike is running safely

Vehicle is running

31) Implementing polymorphism

32) How to do inheritance in java (using extend keyword)

class Calculation {

int z;

public void addition(int x, int y) {

z = x + y;

System.out.println("The sum of the given numbers:"+z);

}

public void Subtraction(int x, int y) {

z = x - y;

System.out.println("The difference between the given numbers:"+z);

}

}

public class My\_Calculation extends Calculation {

public void multiplication(int x, int y) {

z = x \* y;

System.out.println("The product of the given numbers:"+z);

}

public static void main(String args[]) {

int a = 20, b = 10;

My\_Calculation demo = new My\_Calculation();

demo.addition(a, b);

demo.Subtraction(a, b);

demo.multiplication(a, b);

}

}

\*Compile and execute the above code as shown below.

javac My\_Calculation.java

java My\_Calculation\*

o/p: The sum of the given numbers:30

The difference between the given numbers:10

The product of the given numbers:200

|  |  |  |
| --- | --- | --- |
| 33) write code to add items to integer, string, array | |  |
| Packagestrings; | |
|  | |  | | |
|  | | public class UsingString { | | |
|  | | public static void main(String[] args) { | | |
|  | | String value = "Ravali"; // gets already created value | | |
|  | | // there is no append method. | | |
|  | | value = value+" Saibersys"; //creates a new reference in the heap as it is immutable | | |
|  | | System.out.println("using String: "+value); | | |
|  | | StringBuffer value1 = new StringBuffer("appended "); | | |
|  | | value1.append("using append method"); // mutable | | |
|  | | System.out.println(" using string Buffer: "+value1); | | |
|  | |  | | |
|  | |  | | |
|  | | // using StringBuilder | | |
|  | | StringBuilder value2 = new StringBuilder("appended "); | | |
|  | | value2.append("using append method "); | | |
|  | | System.out.println("using StringBuilder: "+value2); | | |
|  | | } | | |
|  | |  | | |
|  | | }  o/p: using String: Ravali Saibersys  using string Buffer: appended using append method  using StringBuilder: appended using append method | | |

34) write code to retrieve items from integer, string array

**package** com.strings;

**public** class Testarray{

public static void main(String args[]){

int a[]=new int[5];//declaration and instantiation

a[0]=10;//initialization

a[1]=20;

a[2]=70;

a[3]=40;

a[4]=50;

//printing array

for(int i=0;i<a.length;i++)//length is the property of array

System.out.println(a[i]);

}}

o/p:

10

20

70

40

50

35) write code to retrieve items from arraylist (using for each loop)

**package** com.strings;

public class TestArray1 {

public static void main(String[] args) {

double[] myList = {1.9, 2.9, 3.4, 3.5};

// Print all the array elements

for (int i = 0; i < myList.length; i++) {

System.out.println(myList[i] + " ");

}

// Summing all elements

double total = 0;

for (int i = 0; i < myList.length; i++) {

total += myList[i];

}

System.out.println("Total is " + total);

// Finding the largest element

double max = myList[0];

for (int i = 1; i < myList.length; i++) {

if (myList[i] > max) max = myList[i];

}

System.out.println("Max is " + max);

}

}

o/p:

1.9

2.9

3.4

3.5

Total is 11.7

Max is 3.5

|  |
| --- |
| 36)HashMap import java.util.HashMap; |
|  | import java.util.Iterator; |
|  | import java.util.Map; |
|  | import java.util.Scanner; |
|  | import java.util.Set; |
|  |  |
|  | public class usingHashmap { |
|  | static HashMap< String, Integer> StudentDetails = new HashMap<String, Integer>(); // can store the key and value |
|  |  |
|  | static void addToArray(String stdName, int rollNumber){ |
|  | StudentDetails.put( stdName, rollNumber); |
|  | } |
|  |  |
|  | /\* To fetch the values from hashmap the following steps are required |
|  | Step 1:Get the set of key-value pairs by calling entrySet() method |
|  | Step:2 obtaining iterator for the key-value pairs |
|  | Step:3 use Map.Entry interface to fetch the key and value\*/ |
|  |  |
|  | static void printStdDetails() |
|  | { |
|  | Set set = StudentDetails.entrySet(); // getting key-value pairs |
|  | Iterator i = set.iterator(); // obtaining iterator for the key-value pairs |
|  | while(i.hasNext()) |
|  | { |
|  | Map.Entry keyValue= (Map.Entry)i.next(); |
|  | System.out.println("StudentName: "+keyValue.getKey()); |
|  | System.out.println("Student RollNumber: "+keyValue.getValue()); |
|  | } |
|  |  |
|  | } |
|  | public static void main(String[] args) { |
|  | Scanner scanner = new Scanner(System.in); // reads value from the console. import from java.util.Scanner; |
|  | for(int i =0; i<5; i++){ |
|  | System.out.println("Please enter student Name: "); |
|  | String stdName =scanner.nextLine(); |
|  | addToArray(stdName, i); //calling method to add |
|  | } |
|  |  |
|  | printStdDetails(); |
|  | System.out.println("\*\*\*\*\*\*\*\*\*End\*\*\*\*\*\*"); |
|  | } |
|  |  |
|  | }  O/p: Please enter student Name:  Ravali  Please enter student Name:  Ravali  Please enter student Name:  Ravali  Please enter student Name:  Ravali  Please enter student Name:  Ravali  StudentName: Ravali  Student RollNumber: 4  \*\*\*\*\*\*\*\*\*End\*\*\*\*\*\* |

37)HashSet:

import java.util.HashSet;

import java.util.Iterator;

public class IterateThroughElementsOfHashSetExample {

  public static void main(String[] args) {

    //create object of HashSet

    HashSet hSet = new HashSet();

    //add elements to HashSet object

    hSet.add(new Integer("1"));

    hSet.add(new Integer("2"));

    hSet.add(new Integer("3"));

    //get the Iterator

    Iterator itr = hSet.iterator();

    System.out.println("HashSet contains : ");

    while(itr.hasNext())

      System.out.println(itr.next());

  }

}

/\*

Output would be

HashSet contains :

3

2

1

|  |  |
| --- | --- |
| 38)write code to connect to JDBC to get rows from employee table  39)Create employee class |  |

40) Difference between string, string buffer, string builder with example

**String**  
  
String is immutable  ( once created can not be changed )object  . The object created as a String is stored in the  Constant String Pool  .   
Every immutable object in Java is thread safe ,that implies String is also thread safe . String can not be used by two threads simultaneously.  
String  once assigned can not be changed.  
  
String  demo = " hello " ;  
// The above object is stored in constant string pool and its value can not be modified.

demo="Bye" ;     //new "Bye" string is created in constant pool and referenced by the demo variable              
 // "hello" string still exists in string constant pool and its value is not overrided but we lost reference to the  "hello"string    
 **StringBuffer**  
  
StringBuffer is mutable means one can change the value of the object . The object created through StringBuffer is stored in the heap . StringBuffer  has the same methods as the StringBuilder , but each method in StringBuffer is synchronized that is StringBuffer is thread safe .   
  
Due to this it does not allow  two threads to simultaneously access the same method . Each method can be accessed by one thread at a time .  
  
But being thread safe has disadvantages too as the performance of the StringBuffer hits due to thread safe property . Thus  StringBuilder is faster than the StringBuffer when calling the same methods of each class.  
  
StringBuffer value can be changed , it means it can be assigned to the new value . Nowadays its a most common interview question ,the differences between the above classes .  
String Buffer can be converted to the string by using   
toString() method.  
  
StringBuffer demo1 = new StringBuffer("Hello") ;  
// The above object stored in heap and its value can be changed .

demo1=new StringBuffer("Bye");  
// Above statement is right as it modifies the value which is allowed in the StringBuffer  
  
**StringBuilder**  
  
StringBuilder  is same as the StringBuffer , that is it stores the object in heap and it can also be modified . The main difference between the StringBuffer and StringBuilder is that StringBuilder is also not thread safe.   
StringBuilder is fast as it is not thread safe .    
StringBuilder demo2= new StringBuilder("Hello");  
// The above object too is stored in the heap and its value can be modified  
demo2=new StringBuilder("Bye");   
// Above statement is right as it modifies the value which is allowed in the StringBuilder

**StringBuffer Example**

public class BufferTest{

    public static void main(String[] args){

        StringBuffer buffer=new StringBuffer("hello");

        buffer.append("java");

        System.out.println(buffer);

    }

}

o/p: hellojava

**StringBuilderExample**

public class BuilderTest{

    public static void main(String[] args){

        StringBuilder builder=new StringBuilder("hello");

        builder.append("java");

        System.out.println(builder);

    }

}

o/p: hellojava