Java:

1.how to import source code from outside?

Extract all the files from mail to specific location in any drive say here I have saved it in Siddharth\_java->Training.

Then select them all and copy them all.

Now go to source folder of Eclipse,say here D->Shruti->Eclipse->Java\_Tutorial\_1->src, and paste all those folders here.

Now open eclipse and do refresh there,all these packages will get reflected there.

Day\_1:

* **CLASS & OBJECT:**

1. Class contains data & methods.
2. It’s a blueprint of an object.
3. It doesn’t exist really.
4. While ‘Object’ exists really.

e.g.

Dog is a class;

Now for data say all the characteristics of a Dog,like age,color,height,breed……

And for methods take behavior of a Dog like barking,running,bitting.

* Now create a class having methods & data and call it Dog.
* Create another class having main method in it which is called a driver method and call it a Dog\_Driver.
* Now,in order to access methods of Dog class we need to create its object in driver class.
* So,after creating object of Dog class in Driver class we can access all its methods.

2.Scope of the variable:

1.All variables which are present inside any loop or for specific method are all having **Local** scope.

2.All the variables which are not declared as private or anything and are not inside any method or a loop are all **Global** variables.

Wrapper classes:

Wrapper classes are used to convert a String to an integer or  
 String to Double.

But reverse of this is not possible.

**JVM vs JDK vs JRE:**

**JDK:**

It’s a superset which is used for ‘Running+Development’.

**JDK** consists of JRE & JVM.Its is used for developing and running a programe.

**JRE:**

Its used for running java programs.

If someone wants to only run java programs then he needs to have JRE installed in his system.  
Java provides JDK and JRE separately.so in case if someone doesn’t want to go for development then he can only go for JRE.no need to get JDK installed.:)

JVM:

JVM consists of all compilers and all like **Javac** file .

Eclipse is an IDE.

**All the executable files are in the form of .exe.**

**OOPS Concepts:**

**1.INHERITANCE:**

**(Inheritance simply means inheriting from parent or from main classes.)**

**I) INTERFACE (contract):**

Interface is created for synchronization.Like,MS Office.there are various programs of MS Office like msWord,power point,excel.Now,in all these when we open ‘file’ it shows very similar options like save,open,saveall,print and so on.

Now for such programing we use ‘Interface’.

In Interface all the methods will be blank.as per the requirement user will define those methods i.e,user will write body as per their requirement. Suppose user wants ‘open’ function of “word” &”excel” to be different.so in main class while implementing Interface user will define their methods as per their requirements.

Interface is **Implemented**.

e.g,

public interface office{

void open();

void save();

void print();

}

Public class word{

Public void open()

{

System.out.println(“open the word file ”);

}

Public void save()

{

System.out.println(“save the word file ”);

}

}

}

Public class excel {

Public void open()

{

System.out.println(“open the excel file ”);

}

Public void save()

{

System.out.println(“save the excel file ”);

}

}

}

Here we have created one interface in which all the methods are empty so that users of excel and users of word can define them as per their need.

This is what interface is used for.

**Abstract classes:**

In abstract classes few methods will be blank while few are defined.

-Why to use abstract classes over an interface?

For an instance take school management system.

Here we know that for different schools users will use different databases.so the methods related to DB operations will differ for them while rest methods will remain same.so all methods related to DB are left blank while rest will be defined.so this is the reason after using Abstract classes.

Abstract classes are extended.

***If there is any abstract method in a class, that class must be abstract.***

#### *Rule: If you are extending any abstract class that have abstract method, you must either provide the implementation of the method or make this class abstract.*

* ***Interface can be implemented by Abstract classes but Abstract classes can’t be extended by Interface.***
* ***(the reason behind this is,interface needs to have all unimplemented methods,i.e.methods not having body.While,Abstract classes might not be having all unimplemented methods.)***

e.g.1:

1. //example of abstract class that have method body
2. **abstract** **class** Bike{
4. Bike(){System.out.println("bike is created");}
6. **abstract** **void** run();
8. **void** changeGear(){System.out.println("gear changed");}
10. }
12. **class** Honda **extends** Bike{
14. **void** run(){System.out.println("running safely..");}
16. }
18. **class** TestAbstraction2{
20. **public** **static** **void** main(String args[]){
21. Honda obj = **new** Honda();
22. obj.run();
24. obj.changeGear();
25. }
26. }

O/P:

bike is created

running safely..

gear changed---this is because when child extends parent class all the methods in parent class will be present in the child class too.

Interface + Abstract classes:

1. **interface** A{
2. **void** a();
3. **void** b();
4. **void** c();
5. **void** d();
6. }
8. **abstract** **class** B **implements** A{
9. **public** **void** c(){System.out.println("I am C");}
10. }
12. **class** M **extends** B{
13. **public** **void** a(){System.out.println("I am a");}
14. **public** **void** b(){System.out.println("I am b");}
15. **public** **void** d(){System.out.println("I am d");}
16. }
18. **class** Test5{
19. **public** **static** **void** main(String args[]){
20. A a=**new** M();
21. a.a();
22. a.b();
23. a.c();
24. a.d();
25. }}

O/P:

Output:I am a

I am b

I am c

I am d

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**POLYMORPHISM:**

**(Polymorphism simply means same names and different usages)**

* **Overloading:**

Overloading means methods with same names and their arguments will vary in their type or number.

Overloading happens in the same class.

It depends on the compiler that when to use which method depending upon the input taken.

* **Overriding:**

For overriding suppose we have one parent class and one child class.

The child class is inheriting the parent class.

Now these both the classes are having same methods but their bodies are different.

So the final content that will be considered is of a child class.

**ENCAPSULATION:**

Encapsulation is integrating complex modules in one.

In Encapsulation we make variables private so that we can keep the data secure and no one from outside will be able to access these variables.

generate their getters & setters to access these variables. Hence by doing this we collect many things inside one thing and hide their complexity from the user.

**TRY-CATCH-FINALLY:**

Exceptions are runtime errors which compiler is unable to find during compilation.they are handled by Try-Catch-Finally blocks.

The operation which we need to perform will be placed in a **TRY** block.

While the exception will be handled in the **CATCH** block.

And **FINALLY** block will always be executed.

Here default exceptions are handled like Arithmetic exception.

In order to create our own exceptions we need to go for “THROWS & THROW”.

**Static Variable**:

The variables declared as a static variable will remember their previous value.i.e.they don’t forget the value assigned to them.

For e.g,

We have two variables

One is static and second is a normal variable.

Like ;

Static int a=0;

Int b=0;

Now suppose these variables are getting incremented thrice by 1.so the value of ‘a’ will be 3 after final increment while that of ‘b’ will be 0.

This happens because after each increment ‘a’ remembers its value while ‘b’ forgets it and always starts from 0.

**FINAL**:

\* Overriding: When parent and child classes has exactly the same method name preference is given to child method.

\* We can stop overriding of the class and variables by using final keyword.final can used with classes, methods and data types.

\* Features of final keyword:

\* 1.You can't inherit final class.

\* 2.You can't override final methods.

\* 3.You can't change final variables.

**Getting an error like JVM cant be created:**

C:\Program Files\Java\jdk1.7.0\bin:

Go to the path where jdk’s bin file is located.

Copy that path as shown above.

Now go to my computer and right click it.->advanced system settings->environment variables->new->variable name=Path->variable value=paste that url.

Now, open cmd window and type javac.

If it opens up then jvm created.!!!

JDBC:

(Java database connectivity):

For JDBC the main thing needed is “Driver”.

Driver

DB

.java

What Driver does is;

It uses code from application applying sql querries & processes data in the Database and brings the result to the Application back.

2 PACKAGES REQUIRED FOR jdbc;

Import.java.sql.\*;

Import.javax.sql.\*;

Main 4 steps of JDBC:

1.Connection to the DB.

This connection is done with using ‘username’ & ‘password’.

2.Statement Object

Statement st=new Statement();

St.executeQuery(“-----------------”);

3.Running sql querry.

4.Result set.

Result set fetches the output of the query in a row.

Main operations of JDBC:

Insert

Update

Delete

Select

Stored Procedure-Used to do something automatically

BL(OB)-Binary large Objest

CL(OB)-Character large object