**Method in Practical example:**

**public** **class** Methodss {

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

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

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

}

}

**1.Pre-defined methods**

* **“Test”** – here Test is an argument or parameter
* Here System and Println is a predefined method which means this method written by the ORACLE which owns java
* When control shift selected over System or Println it will open a new class

**2.User-defined methods**

**public** **class** Methodss {

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

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

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

*simplemethod*(); // here we calling the method

}

**public** **static** **void** simplemethod()

{

System.***out***.println("Simply super machi");

}

}

**Accepting the arguments when calling a method:**

When giving arguments when calling a method it will not accept it throws **“the method simplemethod() in the type Methodss is not applicable for the arguments (String)”””**

**public** **class** Methodss {

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

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

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

*simplemethod*("super");

}

**public** **static** **void** simplemethod()

{

System.***out***.println("Simply super machi");

}

}

* **Because in** simplemethod() are not defined ,so we need to define it

**How we define it?**

* simplemethod(String arguments)

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

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

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

*simplemethod*("simply awesome da");

}

**public** **static** **void** simplemethod(String arguments)// we defining the aruguments as String

{

System.***out***.println("Simply super machi");

}

}

* Here arguments is a variable name of String data type which is assigned to "simply awesome da” string
* So with above we can do various thing

**How to Append variable in SYSO output**

**public** **class** Methodss {

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

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

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

*simplemethod*("simply awesome da");

}

**public** **static** **void** simplemethod(String arguments)// we defining the aruguments as String

{

System.***out***.println("Simply super machi" + **arguments** );

}

}

Here we added variable in syso,so it will append in output

So the output will be

Test

Simply super machi simply awesome da

Note : if we change the parameter output will show as per the change

Example - *simplemethod*("simply awesome da"); we can change TO - *simplemethod*("Hi I am Nazeer");

Imagine if u want to print integer it will throw “this method is not applicable for the arguments”

**public** **class** Methodss {

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

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

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

***simplemethod*(34); //** he method simplemethod(String) in the type Methodss is not applicable for the arguments (int)

}

**public** **static** **void** simplemethod(String arguments)// we defining the aruguments as String

{

System.***out***.println("Simply super machi" + arguments );

}

}

Above code we defined as data as String but we are passing as INTEGER,so it illegal so it throws error

So how we going to solve it?

**public** **class** Methodss {

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

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

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

*simplemethod*(34);

}

**public** **static** **void** simplemethod(String arguments)// we defining the aruguments as String

{

System.***out***.println("Simply super machi" + arguments );

}

**public** **static** **void** simplemethod(**int** arguments)//

{

System.***out***.println("Simply super machi" + arguments );

}

}

Here we created another method with same method name

Now we give *simplemethod*(34); as Integer datatype

Now we have 2 method which will accept both string and INT

So java intelligently take INT variable of 2nd method and assign it on the *simplemethod*(34); when we call it

So to conform it is running 2nd method is running

**public** **static** **void** simplemethod(**int** arguments)//

{

System.***out***.println("integer is running" + arguments );

}

}

OUTPUT – Integer is running 34

Now we gave quotation it will take as String

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

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

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

*simplemethod*("34");

}

**public** **static** **void** simplemethod(String arguments)// we defining the aruguments as String

{

System.***out***.println("String is running" + arguments );

}

**public** **static** **void** simplemethod(**int** arguments)//

{

System.***out***.println("Simply super machi" + arguments );

}

OUTPUT - String is running 34

**We going to define the method in Another class:**

**1.Create a class (Myutils.java) with Methodssss class method like below**

**public** **static** **void** simplemethod(String arguments)// we defining the aruguments as String

{

System.***out***.println("String is running" + arguments );

}

**public** **static** **void** simplemethod(**int** arguments)//

{

System.***out***.println("Simply super machi" + arguments );

Methodsss.java will throw error

simplemethod("34"); // The method simplemethod(String) is undefined for the type Methodss

Why it showed this error, because now the methods implemented in **Myutils.java to resolve we need to do following**

**Myutils.** simplemethod("34"); // Here simplemethod("34") is defined in Mytils.java class

**public** **class** Methodss {

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

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

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

Myutils.*simplemethod*("34");

}

}

**We can create another method in Muutils.java class**

**public** **static** **void** sum2numbers(**int** firstarg , **int** secondarg)

{

System.***out***.println(firstarg + secondarg);

}

**Here** Myutils.*sum2numbers*(10,20); (10,20) is ARGUMENTS

**int** firstarg , **int** secondarg Firstarg,secondarg is parameter

An argument is referred to the values that are passed within a method when the method is called

The parameter is referred to as the variables that are defined during a function declaration or definition.

Parameter values can be changed bcz it is variables

Don’t worry if u get confused between ARGUMENTS and Paramter

**public** **class** Methodss {

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

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

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

Myutils.*simplemethod*("34");

Myutils.*sum2numbers*(10,20);

}

**We going to create another method to add10 with Return**

**public** **static** **int** add10(**int** someargument) // we need to give int datatype to add10

{

**int** result=someargument + 10;

**return** result;

//when we put return we need to mention returntype here int is a return type

}

}

**package** learns;

**public** **class** Methodss {

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

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

//Myutils.add10(99); Now if we run this nothing will get return becz we are not saving the return datatype,we need to save it

**int** myvar=Myutils.*add10*(99); // now again it will NOT return bcz it is just saving the values

**int** myvar=Myutils.*add10*(99);

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

}

// it will return the value bcz of System.***out***.println(myvar);

Now we can give

**int** myvar=Myutils.*add10*(99)+1000;

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

**Access Modifier**

* When Method access level is changed from public to private
* Parent class will not allowed use the method
* Private will only allowed to use on the same class

**Static keyword**

* When method is Static ,no need to create instances or object
* When method is non-static ,we need to create instances

**package** learns;

**public** **class** Myutils {

**public** **int** nonstatic(**int** someargument) // we need to give int datatype to add10

{

**int** result=someargument + 10;

**return** result;

}

**package** learns;

**public** **class** Methodss {

Myutils newvar= **new** Myutils();

newvar.nonstatic(10);