DAY-22

-------

MUTATORS or SETTERS and ACCESSOR or getters

-------------------------------------------

Based on the object state modification methods can be classified into two types:

1.MUTATORS/SETTER : These are used to set or modify the value in the object.

2.ACCESSOR/GETTERS : These are used to get the value from the object.

// example:1 MUTATORS/SETTERS and ACCESSORS/GETTERS

class Credentials

{

private String un;

private String pwd;

void setUn(String username)

{

un=username;

}

void setPwd(String password)

{

pwd=password;

}

String getUn()

{

return un;

}

String getPwd()

{

return pwd;

}

}

class Demo

{

public static void main(String[] args)

{

Credentials c = new Credentials();

c.setUn("StudyOnline");

c.setPwd("3443");

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

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

}

}

output : --> studyOnline

34343

// example:2 MUTATORS/SETTERS and ACCESSORS/GETTERS

class Student

{

private String name;

private int age;

private float height;

void setName(String sname)

{

name=sname;

}

void setAge(int sage)

{

age=sage;

}

void setHeight(float sheight)

{

height=sheight;

}

String getName()

{

return name;

}

int getAge()

{

return age;

}

float getHeight()

{

return height;

}

}

class Demo

{

public static void main(String[] args)

{

Student s = new Student();

s.setName("Subbu");

s.setAge(24);

s.setHeight(6.1f);

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

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

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

}

}

output: --> subbu

24

6.1

PASS BY VALUE

-------------

In java the parameter can be passed only in the value mode. Java doesnot support the any other passing techniques other than pass by value.

In c programming language it has both pass by value and pass by reference, but java doesnot support concept of pointer hence pass by refernece is

ruled out in java.

// PASS BY VALUE IN JAVA.

--------------------------

class Test

{

void swap(int x, int y)

{

int temp;

temp = x;

x = y;

y = temp;

}

}

class Demo1

{

public static void main(String[] args)

{

int a=10;

int b=20;

Test t = new Test();

System.out.println("before swapping:");

System.out.println("the value of a is :" + a);// 10

System.out.println("the value of b is :" + b);// 20

t.swap(a,b);

System.out.println("after swapping:");

System.out.println("the value of a is :" + a); // 20

System.out.println("the value of b is :" + b);// 10

}

}

output:

-------

before swapping:

the value of a is :10

the value of b is :20

after swapping:

the value of a is :10

the value of b is :20

NOTE:Whatever the changes made in called method is reflected on calling method.

METHOD OVERLOADING IN JAVA

--------------------------

In c language we can write any number of functions but the functions name should be unique(no two functions shuold have the same name).

example of c language:

----------------------

#include<stdio.h>

#include<conio.h>

Void main()

{

Int a=10, b=20, c=30;

float x=1.2, y=2.2, z=3.2;

double p=10.222, q=20.2433, r=30.3233;

printf(“%f”, add3(p,q));

printf(“%f”, add4(a,x));

printf(“%f”, add10(a,b,c));

int add1(int x, int y)

{

return x+y;

}

int add2(float x, float y)

{

return x+y;

}

int add3(double x, double y)

{

return x+y;

}

int add4(int x, float y)

{

return x+y;

}

int add5(int x, double y)

{

return x+y;

}

int add6(float x, double y)

{

return x+y;

}

int add7(float x, int y)

{

return x+y;

}

int add8(double x, int y)

{

return x+y;

}

int add9(double x, float y)

{

return x+y;

}

int add10(int x, int y, int z)

{

return x+y+z;

}

int add11(float x, float y, float z)

{

return x+y+z;

}

int add12(double x, double y, double z)

{

return x+y+z;

}

int add13(int x, float y, double z)

{

return x+y+z;

}

int add14(int x, float y, float z)

{

return x+y+z;

}

int add15(int x, double y, double z)

{

return x+y+z;

}

int add16(float x, int y, int z)

{

return x+y+z;

}

note:

-----

In the above c program even though we are properly performing addition operation but still we should remember multiple functions name which is complex

The above problem can be overcome using the METHOD OVERLOADING concept in java.

example:

------------

// METHOD OVERLOADING IN JAVA

class Demo2

{

public static void main(String[] args)

{

Test t = new Test();

int a=10, b=20, c=30; // NAME OF THE METHOD

float x=1.2f, y=2.2f, z=3.2f; // NUMBER OF PARAMETER

double p=10.222, q=20.2433, r=30.3233; // DATATYPES OF THE PARAMETER

// ORDER OF THE PARAMETER

System.out.println(t.add(p,q));

System.out.println(t.add(a,x));

System.out.println(t.add(a,b,c));

System.out.println(t.add(a,p));

}

}

class Test

{

int add(int x, int y)

{

return x+y;

}

float add(float x, float y)

{

return x+y;

}

double add(double x, double y)

{

return x+y;

}

float add(int x, float y)

{

return x+y;

}

double add(int x, double y)

{

return x+y;

}

double add(float x, double y)

{

return x+y;

}

float add(float x, int y)

{

return x+y;

}

double add(double x, int y)

{

return x+y;

}

double add(double x, float y)

{

return x+y;

}

int add(int x, int y, int z)

{

return x+y+z;

}

float add(float x, float y, float z)

{

return x+y+z;

}

double add(double x, double y, double z)

{

return x+y+z;

}

double add(int x, float y, double z)

{

return x+y+z;

}

float add(int x, float y, float z)

{

return x+y+z;

}

double add(int x, double y, double z)

{

return x+y+z;

}

float add(float x, int y, int z)

{

return x+y+z;

}

}

output:

-------

30.4653

11.2

60

20.222

Multiple methods with the same name is called as method overloading . Whenever call is made for overloaded method the compiler will resolve the

method call in following ways:

1.NAME OF THE METHOD

2.NUMBER OF PARAMETER

3.DATATYPES OF THE PARAMETER

4.ORDER OF THE PARAMETER