Object Oriented Programming
CS F213

J. Jennifer Ranjani

email: jennifer.ranjani@pilani.bits-pilani.ac.in

Chamber: 6121 B, NAB

Consultation: Appointment by e-mail

BITS Pilani

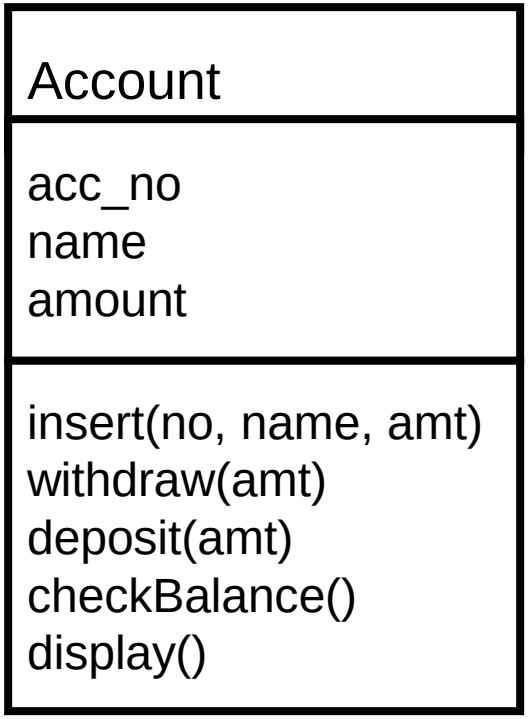
Pilani Campus

Classes, Methods, Constructor & Overloading

BITS Pilani
Pilani Campus

Class & Object - Example

- Write a java program for the class diagram given below.



```
class Account{
int acc_no;
String name;
float amount;
void insert(int a,String n,float amt){
acc_no=a;
name=n;
amount=amount+amt;
}
void deposit(float amt){
amount=amount+amt;
System.out.println(amt+" deposited");
}
void withdraw(float amt){
if(amount<amt){
System.out.println("Insufficient Balance");
}else{
amount=amount-amt;
System.out.println(amt+" withdrawn");
}
 }
```

```
void checkBalance(){System.out.println("Balance is: "+amount);}
void display(){System.out.println(acc_no+" "+name+" "+amount);}
}
class TestAccount{
public static void main(String[] args){
Account a1=new Account();
a1.insert(832345,"Ankit",1000);
a1.display();
a1.checkBalance();
a1.deposit(40000);
a1.checkBalance();
a1.withdraw(15000);
a1.checkBalance();
}}
```

Method Overloading

- Multiple methods having same name but different parameters is known as method overloading.

- Eg. Suppose you have to perform addition of the given numbers but there can be any number of arguments, if you write the method such as a(int,int) for two parameters, and b(int,int,int) for three parameters then it may be difficult for you as well as other programmers to understand the behavior of the method because its name differs.

- Adv: increases the readability of the program.

Different ways to overload

- Changing the number of arguments
  - **int** add(**int** a,**int** b)
  - **int** add(**int** a,**int** b,**int** c)
- Changing the data type
  - **int** add(**int** a, **int** b)
  - **double** add(**double** a, **double** b)
- Changing only the return type does not mean method overloading
  - **int** add(**int** a,**int** b)
  - **double** add(**int** a,**int** b)
  - Compile Time Error: method add(int,int) is already defined in class Adder

Method Overloading - Example

innovate achieve lead

```
class Account{
int acc_no;
String name;
float amount;
void insert(int a,String n,float amt){
acc_no=a;
name=n;
amount=amt;  }
void insert(int a,String n){
acc_no=a;
name=n;
amount=1000;  }
void display(){
System.out.println(acc_no+" "+name+" "+amount);}
}
```

Minimum balance is 1000

Method Overloading - Example

innovate achieve lead

```
class TestAccount{
public static void main(String[] args){
Account a1 =  new Account();
a1.insert(832345,"Ankit",5000);
a1.display();

Account a2 =  new Account();
a2.insert(832346,“Shobit“);
a2.display();
}}
```

Output:
832345 Ankit 5000.0
832346 Shobit 1000.0

Can Main() be overloaded?

innovate achieve lead

```
public static void main(String[] args){System.out.println("main with String[]");}
public static void main(String args){System.out.println("main with String");}
public static void main(){System.out.println("main without args");}
```

Ans: Yes. JVM calls main() method which receives **string array** as arguments only.

Overloading and Type Promotion


```
class OverloadingCalculation{
  void add(int a,long b){System.out.println(a+b);}
  void add(int a,int b,int c){System.out.println(a+b+c);}

  public static void main(String args[]){
  OverloadingCalculation obj=new OverloadingCalculation();
  obj.add(20,20);
  obj.add(20,20,20);
  }
}
```

Output:
40
60

Overloading and Type Promotion (Matching Type Arguments)


```
class OverloadingCalculation{
  void add(int a,int b){System.out.println("int arg method invoked");}
  void add(long a,long b){System.out.println("long arg method invoked");}


  public static void main(String args[]){
  OverloadingCalculation obj=new OverloadingCalculation();
  obj.add(20,20);
  }
}
```

Output:
int arg method invoked

Overloading and Type Promotion (Ambiguity)

innovate achieve lead

```
class OverloadingCalculation{
  void add(int a,long b){System.out.println("a method invoked");}
  void add(long a,int b){System.out.println("b method invoked");}

  public static void main(String args[]){
  OverloadingCalculation obj=new OverloadingCalculation();
  obj.add(20,20);

  }
}
```

Output:
Compile time error

Constructors

- Similar to a method but it is called when an instance of the object is created and memory is allocated for the object.
- Used to initialize an object
- Constructor – constructs values at the time of object creation. It is not necessary to write a constructor for a class, the compiler creates a default constructor.

More about constructors

Rules for creating constructor

- Name must be same as its class name
- Must have no explicit return type.

Types of constructors

- Default (no argument) constructor
  - Provide default values to the object like 0, null etc.
- Parameterized constructor
  - Provide different values to distinct objects.

Default Constructor - Example

innovate achieve lead

```
class Account{
int acc_no;
String name;
float amount;
void display(){
System.out.println(acc_no+" "+name+" "+amount);}
}

class TestAccount{
public static void main(String[] args){
Account a1=new Account();
a1.display();
}}
```

Output:
0 null 0.0

Default Constructor - Example

```
class Account{
int acc_no;
String name;
float amount;
Account(){
System.out.println("The default values are:");
amount = 1000;
}
void display(){
System.out.println(acc_no+" "+name+" "+amount);}
}

class TestAccount{
public static void main(String[] args){
Account a1=new Account();
a1.display();
}}
```

Minimum balance is 1000

Output:
The default values are:
0 null 1000.0

Parameterized Constructor- Example

innovate achieve lead

```
class Account{
int acc_no;
String name;
float amount;
/*void insert(int a,String n,float amt){
acc_no=a;
name=n;
amount=amt;  */
Account(int acc,String aname, float amt){
acc_no = acc;
name = aname;
amount = amt;  }
void display(){
System.out.println(acc_no+" "+name+" "+amount);}
}
```

Parameterized Constructor- Example

innovate achieve lead

```
class TestAccount{
public static void main(String[] args){
/* Account a1 =  new Account();
a1.insert(832345,"Ankit",5000);
a1.display(); */
Account a1=new Account(832345,"Ankit",5000);
a1.display();
 }}
```

Output:
832345 Ankit 5000.0

Note: When parameterized constructors are implemented; then the copy of the default constructor is not created. In this example you cannot create the object as

Account a1 = new Account();

Constructor Overloading

- **Recall: Constructor is just like a method but without return type.**
- **Constructor overloading:** Having more than one constructor with different parameter lists.
- The compiler differentiates by the **number of parameters** in the list and their **types**.

Constructor Overloading-Example

innovate achieve lead

```
class Account{
int acc_no;
String name;
float amount;
Account(int acc,String aname){
acc_no = acc;
name = aname;
amount = 1000;   }
Account(int acc,String aname, float amt){
acc_no = acc;
name = aname;
amount = amt;  }
void display(){
System.out.println(acc_no+" "+name+" "+amount);}
}
```

Constructor Overloading-Example


```
class TestAccount{
public static void main(String[] args){
Account a1=new Account(832345,"Ankit",5000);
a1.display();
Account a2=new Account(832346,”Shobit");
a2.display();
 }}
```

Output:
832345 Ankit 5000.0
832346 Shobit 1000.0

Difference between a Constructor and method

Java Constructor	Java Method
Constructor is used to initialize the state of an object.	Method is used to expose behavior of an object.
Constructor must not have return type.	Method must have return type.
Constructor is invoked implicitly.	Method is invoked explicitly.
The java compiler provides a default constructor if you don't have any constructor.	Method is not provided by compiler in any case.
Constructor name must be same as the class name.	Method name may or may not be same as class name.

innovate achieve lead

BITS Pilani

Pilani Campus

Static Keyword

Static Method

- A static method belongs to the class rather than object of a class.
- A static method can be invoked without the need for creating an instance of a class.
- Static method can access static data member and can change the value of it.
- Restrictions
  - Static method cannot use non static data member or call non-static method directly i.e. non static members can only be accessed using objects.
  - this and super keywords cannot be used in static context.

Static Method - Example

innovate achieve lead

```
class Account{
int acc_no;
String name;
float amount;
static int branch;
Account(int acc,String aname, float amt){
acc_no = acc;
name = aname;
amount = amt;  }
static void change(int bch)
{
branch = bch;   }
void display(){System.out.println(acc_no+" "+name+" "+amount+"
    "+branch);}
}
```

Static Method - Example

innovate achieve lead

```
class TestAccount{

public static void main(String[] args)
{
Account.branch = 111;
Account a1=new Account(832345,"Ankit",5000);
// Account.change(222);
a1.display();
Account.change(222);
Account a2=new Account(832346,"Shobit",1000);
a2.display();

  }
}
```

Static block

- Is used to initialize the static data member.
- It is executed before main method at the time of class loading.
- **Example**

```
class first{
static int i;
static{i = 100; System.out.println("static block is invoked with i="+i);}
  public static void main(String args[]){
   System.out.println("Hello main");
  } }
```

Output:
static block is invoked with i=100
Hello main

Questions

- Can we overload static methods?
  - ‘Yes’. We can have two or more static methods with same name, but differences in input parameters.

- Can we overload methods that differ only by static keyword?
  - No.