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 and Objects

BITS Pilani
Pilani Campus

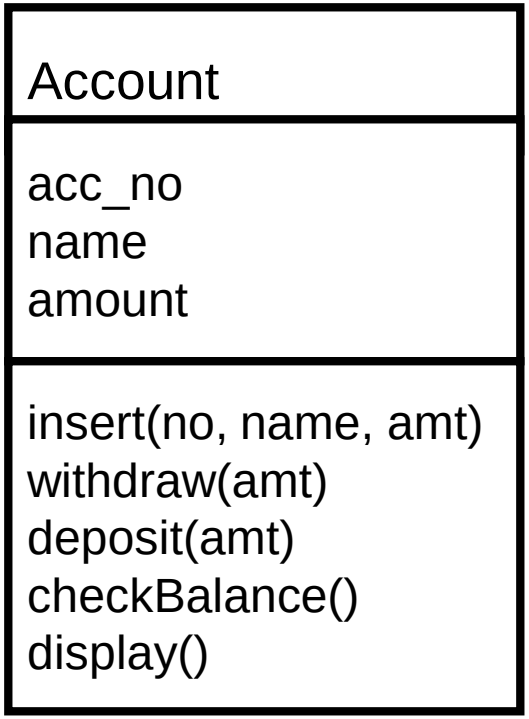
Queries asked During Previous Class

- Can a static (class) variable and local variable have same name. If yes, how can we differentiate if we have to access them both within the method where the local variable is declared.

- Likewise, can an instance and local variable have same name. If yes, how do we handle this?

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=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System.out.println(amt+" withdrawn");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(){display();}
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();
}}
```

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

```
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();

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 will not be same as the class name.

Passing Objects to Methods

- Java is strictly pass by value
- Call by reference can be achieved when objects are passed as arguments
  - When a variable of class type is created, it implies that a reference to an object is created.
    - Eg: Account a1;
  - Reference variable is used to store the address of the object.
  - When the reference is passed to a method, the parameter that receives refer to the same object.

Passing Objects - Example

innovate achieve lead

```
class Account{
int acc;
String name;
float amount;

Account(int act,String aname){
acc = act;
name = aname;
}

boolean equalTo(Account a) {
return(acc == a.acc && name == a.name);
}

}
```

Passing Objects - Example

innovate achieve lead

```
class TestAccount{
public static void main(String[] args){
Account a1=new Account(832345,"Ankit");
Account a2=new Account(832345,"Ankit");
Account a3=new Account(832346,"Shobit");

System.out.println("a1==a2: " + a2.equalTo(a1));
System.out.println("a1==a3: " + a3.equalTo(a1));

}}
```

Output:
a1==a2: true
a1==a3: false

Assigning Object Reference Variables

- Value of a reference variable can to assigned to another reference variable.
- Assigning reference will not create distinct copies of objects.
- All reference variables are referring to the same object.

Assigning Object Reference


```
class Account{
int acc;
String name;
float amount;
Account(int act,String aname){
acc = act;
name = aname;
}

boolean equalTo(Account a) {
return(acc == a.acc && name == a.name);
}
void display(){
System.out.println(acc+" "+name+" "+amount);}
}
```

Assigning Object Reference


```
class second{
public static void main(String[] args){
Account a1=new Account(832345,"Ankit");
Account a2= a1;
Account a3=new Account(832346,"Shobit");

System.out.println("a1==a2:" + a2.equalTo(a1));
System.out.println("a1==a3:" + a3.equalTo(a1));

a1.name="Aankit";
a1.display();
a2.display();
 }}
```

Output:

a1==a2: true
a1==a3: false
832345 Aankit 0.0
832345 Aankit 0.0