Java Fundamentals

1. Access specifiers of a class and Method

- A class has 2 access specifiers.

public & default

class Person{ //default access specifier

- you cannot create an object of this class outside of the package.

}

public class Person{ //public access specifier

- you can create an object of this class outside of the package

}

Access Specifiers of Method

- A method has 4 access specifiers

public

protected

default

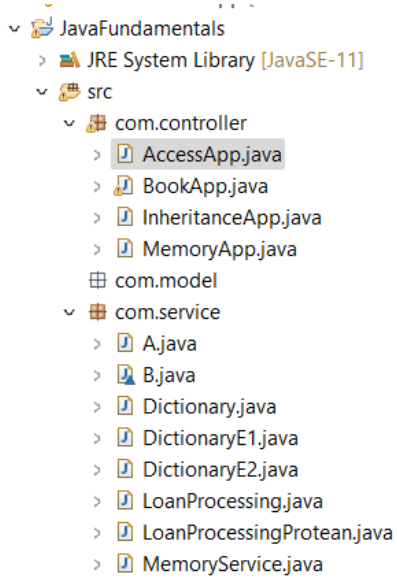
private

if a method is marked as public, it can be called within and outside of its package

if a method is marked as protected, it can be called within the package, but needs to called from an obj of sub-class from outside the package

if a method is marked as default, it can be called within its package, but not outside the package.

if a method is marked as private, it can be called only within its class.



AccessApp.java

```java
package com.controller;

import com.service.A;

public class AccessApp {
    public static void main(String[] args) {
        A a=  new A(); //since A is public
        // B b=new B(); -- Make B public to let controller create its object.
        int sum = a.doSum(4,5); //doSUm in A is public
        System.out.println("sum is : " + sum);

        /*
        int sub = a.doSub(5,6); //-CF doSub in A is not public
        System.out.println("Sub is : " + sub);
        */

        a.setTotalMarks(400);
        a.setMarksScored(340);

        double percent = a.computePercent();

        System.out.println("percent is " + percent);

    }
}
```

A.java

```
package com.service;

public class A {

    int totalMarks;  //instance variable -- 0
    int marksScored; //instance variable -- 0

    public int doSum(int i, int j) {
        return i+j;
    }

    int doSub(int i, int j) { //default method
        return i-j;
    }

    public void setTotalMarks(int totalMarks) { //local variable
        this.totalMarks = totalMarks;  //instance = local
    }

    public void setMarksScored(int marksScored) { //local variable
        this.marksScored = marksScored;

    }

   public double computePercent() {
       double percent = (this.marksScored * 100) / this.totalMarks;
       return percent;
   }

}
```

```
package com.service;

 class B { //default access - can be accessed only within com.service

}
```

2. Local and Instance Variables of the class.

- instance variables can be accessed in all the methods of the class

- local variables can be accessed only within the methods in which they are declared.

Note: If the names of local and instance variables are same, we can use 'this' operator to recognize instance variables.

Example:

3. Working with Object- Understanding Heap and Stack.

- All references are saved on the stack

- All Object go in the heap

Note: Heap is a central memory of the application maintained by JVM.

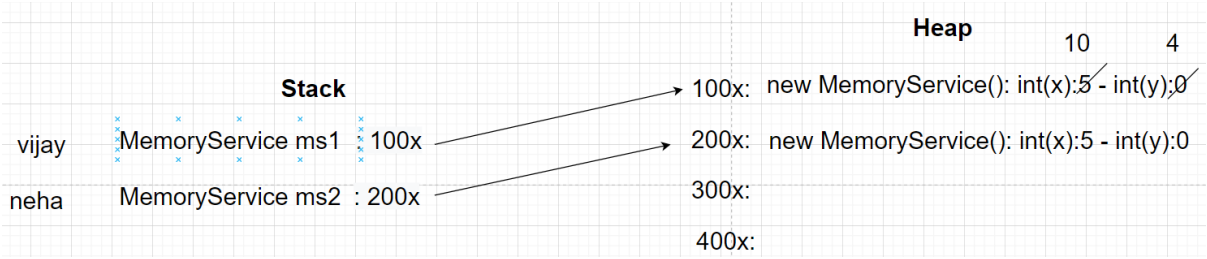
- Once the object is created in the heap, JVM will go to the class and check if there are any instance variables present?

if present: it will save all instance variables along with the object.

Example:-

```java
package com.service;

public class MemoryService {
    private int x=5; //x=5
    private int y;  //y=0

    public void setX(int x) {
        this.x = x;
    }

    public void setY(int y) {
        this.y = y;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }
}
```

```java
package com.controller;

import com.service.MemoryService;

public class MemoryApp {
    public static void main(String[] args) {
        MemoryService ms1 = new MemoryService();
        MemoryService ms2 = new MemoryService();

        ms1.setX(10);
        ms1.setY(4);

        System.out.println(ms1.getX()); //10
        System.out.println(ms2.getX()); //5

        System.out.println(ms1.getY()); //4
        System.out.println(ms2.getY()); //0

        System.out.println(ms1); //123772c4
        System.out.println(ms2); //2d363fb3
    }
}
/*
 * MemoryService ms1 : reference variable (ms1)
 * new MemoryService() : object
 */
```



4. Inheritance & Polymorphism

Extending a class to fetch all its methods into a new class.

example:

---

class A which has 50 methods

As per the new requirement, some methods (could be 5-10) need to be modified.

So the approach should be,

To create a new class say class B{} and extend class A so that all the methods of A come to B and then override only those methods that need modification.

```
class A{ //A has 3 methods
  void m1(){}
  void m2(){}
  void m3(){}
}

class B extends A{
//B also now has 3 methods and B needs to decide which one it needs to override.
}
```

Polymorphism

multiple forms of an object

```
class A{  //super
  void m1(){  m1 of A }
  void m2(){
        -- m2 of A
  }
  void m3(){ m3 of A }
}

class B extends A{ //sub
  //B wants to override m2()
  void m2(){
        -- m2 of B
  }
}
```

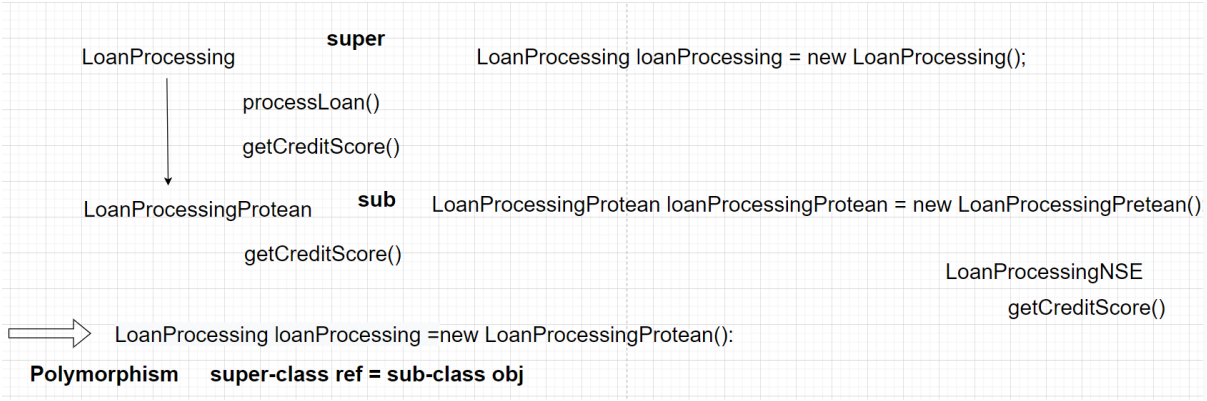
Rule: If a method is called from a polymorphic object,

a. JVM goes to superclass and checks if that method is present or not.

Exam watch: If the method is not present, its a CF(error)

b. JVM goes to sub-class and checks if the method is overrided, if yes, calls overrided method else calls super-class method.

```
A a =new B(); //polymorphic object
a.m1(); //m1 of A
a.m3(); //m3 of A
a.m2(); //m2 of B
```



Example:

```
package com.controller;

import com.service.LoanProcessing;

public class InheritanceApp {
    public static void main(String[] args) {
        LoanProcessing loanProcessing = new LoanProcessingProtean();
        loanProcessing.getCreditScore();
        loanProcessing.processLoan();
        //loanProcessing.processInterest(); -- CF not present in super class
    }
}
```

```
package com.service;

/*
 * Author: Imtiyaz Hirani
 * Date: 15-01-2023
 * */
public class LoanProcessing {

    public void processLoan() {
        System.out.println("process loan using rule 1 and 2");
    }

    public void getCreditScore() {
        System.out.println("Read score from cibil.com");
    }
}

//processBalance() : Account
//getLimit() : CurrentAccount extends Account
//getLimit() : SavingsAccount extends Account
```

```java
package com.service;
/*
 * Author: Imtiyaz Hirani
 * Date: 15-03-2024
 * */
public class LoanProcessingProtean extends LoanProcessing{
    public void getCreditScore() {
        System.out.println("Read score from protean.gov");
    }

//  public void processInterest() {
//      System.out.println("interest processed.");
//  }
}
```

Example No 2: Dictionary

```java
package com.controller;

import com.service.Dictionary;

public class BookApp {
    public static void main(String[] args) {
        Dictionary dictionary = new DictionaryE2();
        dictionary.authorInfo();
        dictionary.abstractInfo();
        dictionary.publishIndex();
    }
}
```

```java
package com.service;

/*
 * Author:
 * date: 4-01-24
 * */
public class Dictionary {

    public void publishIndex() {
        System.out.println("index published as of jan '24");
    }

    public void authorInfo() {
        System.out.println("Author Info");
    }

    public void abstractInfo() {
        System.out.println("abstract Info");
    }
}
```

```java
package com.service;

public class DictionaryE1 extends Dictionary{

    public void publishIndex() {
        System.out.println("index published as of march '24 -- E1");
    }
}
```

```java
package com.service;

public class DictionaryE2 extends DictionaryE1{

    public void publishIndex() {
        System.out.println("index published as of july '24 -- E2");
    }

}
```