Lecture – 7

Static Data and Methods

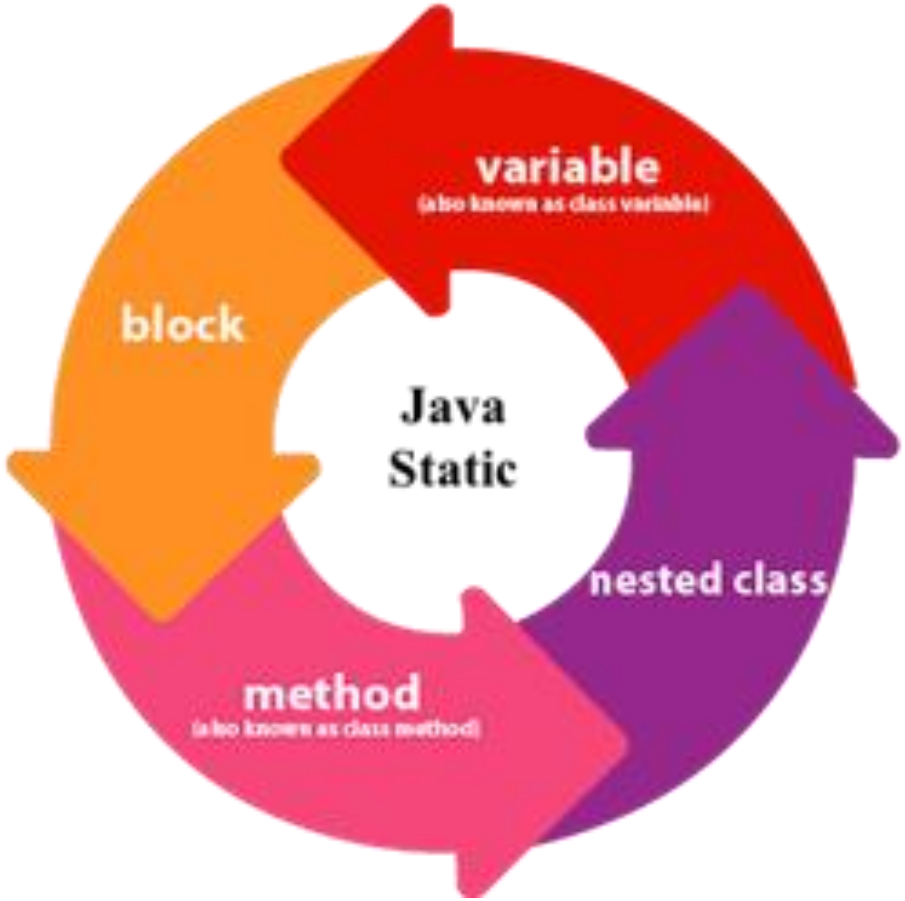
Java static keyword

- Java supports definition of global methods and variables that can be accessed without creating objects of a class. Such members are called **Static members**.
- Define a variable by marking with the static keyword .
- This feature is useful when we want to create a variable common to all instances of a class.
- One of the most common example is to have a variable that could keep a count of how many objects of a class have been created.
- Note: Java creates only one copy for a static variable which can be used even if the class is never instantiated.

Java static keyword

The static can be:

- Variable (also known as a class variable)
- Method (also known as a class method)
- Block
- Nested class



Understanding the Problem without `static`

Suppose there are 500 students in my college, now all instance data members will get memory each time when the object is created. All students have its unique rollno and name, so instance data member is good in such case. Here, "college" refers to the common property of all **objects**. If we make it static, this field will get the memory only once.

```
class Student{
    int rollno;
    String name;
    String college="ITS";
}
```

- `static` is used for a constant variable or a method that is same for every instance of a class.

Example Static

```
class Student{
   int rollno;//instance variable
   String name;
   static String college ="CUI";//static variable
   Student(int r, String n){
     rollno = r;
     name = n;
   }
   Student(){}
   //method to display the values
   void display (){
      System.out.println(rollno+" "+name+" "+college);
   }
}
```

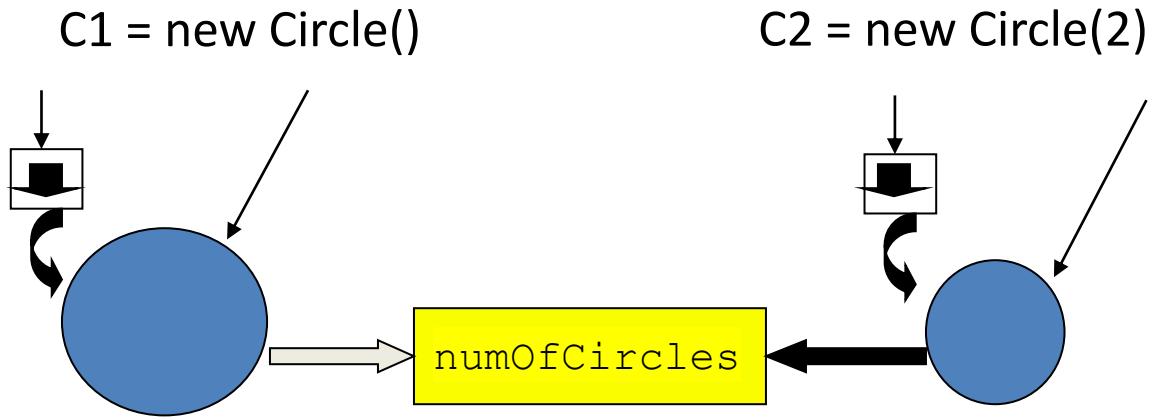
```
public class StudentTest{
  public static void main(String[] args){
     Student s1 = new Student(111,"Ahmad");
      Student s2 = new Student(222,"Zainab");
      s1.display();
      s2.display();

   }
}
```

```
111 Ahmad CUI
222 Zainab CUI
```

Defining Class Variable as Static

```
public class Circle{
    public static int numOfCircles;
    private int radius;
    private final double PI = 3.14;
    public Circle(){
        radius = 0;
        numOfCircles++;
    }
    public void setRadius(int r){
        this.radius = r;
    }
    public Circle(int radius){
        this.radius = radius;
        numOfCircles++;
    }
    public double getArea(){
        return radius * radius * PI;
    }

}
```

```
public class CircleTest{
    public static void main(String[] args){
        Circle c1 =  new Circle();
        c1.setRadius(2);
        System.out.println("C1 Area: "+ c1.getArea());
        Circle c2 = new Circle(4);
        System.out.println("C2 Area: "+ c2.getArea());
        System.out.println("So far we have "+
                Circle.numOfCircles+" circle objects");

    }
}
```



Non-static Vs Static Variables

- Non-static variables : One copy per object. Every object has its own instance variable.
  - E.g. `radius` in the circle

- Static variables : One copy per class.
  - E.g. `numOfCircles` (total number of circle objects created)

Important Points

- *Use a static variable when all objects of a class must use the same copy of the variable.*
- Static variables have class scope. We can access a class's public static members through a reference to any object of the class(`c1.numOfCircles`), or with the class name and a dot (.)(Circle.numOfCircles)

- A class's private static class members can be accessed by client code only through methods of the class.

Static Methods

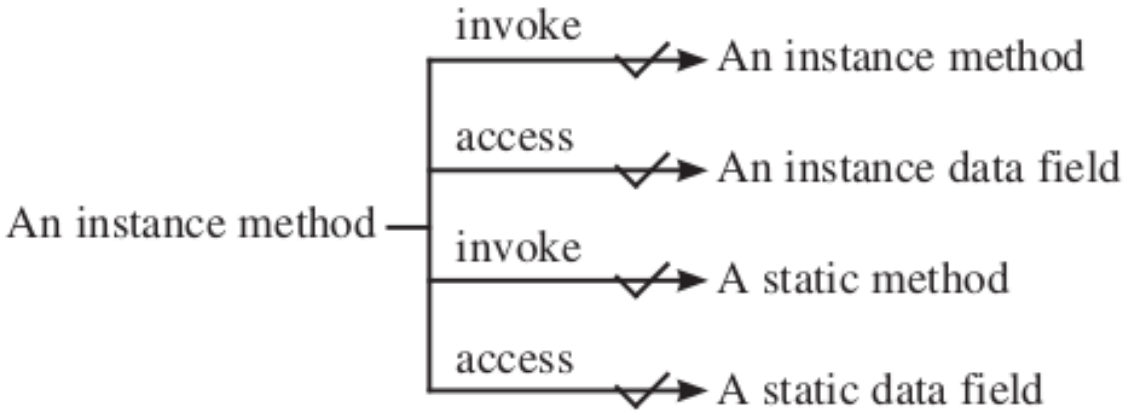
- A static method belongs to the class rather than the object of a class.
- A static method can be invoked without the need for creating an instance of a class.
- A static method can access static data member and can change the value of it.

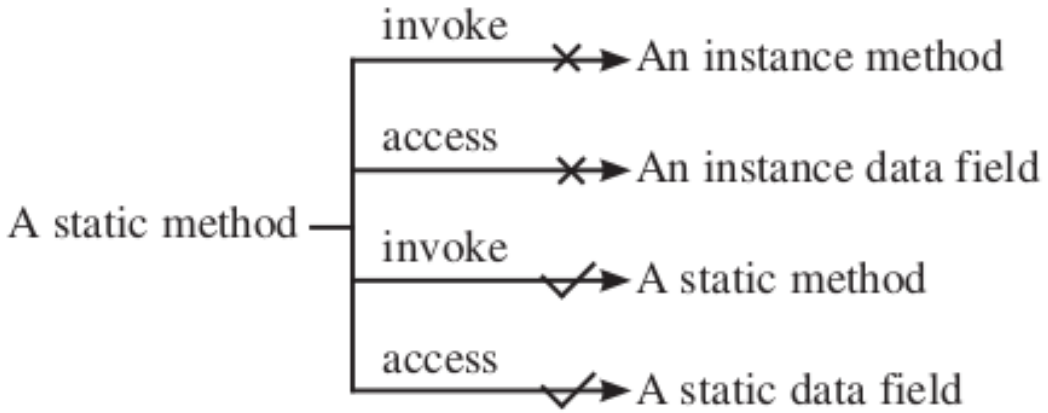
```
class Student{
   int rollno;//instance variable
   String name;
   private static String college ="CUI";//static variable
   Student(int r, String n){
     rollno = r;
     name = n;
   }
   Student(){}
   void display (){
      System.out.println(rollno+" "+name+" "+college);
   }
    public static void changeUni(String uni){
        university = uni;
   }

}
```

```
111 Ahmad QAU
222 Zainab QAU
```

```
public class StudentTest{
  public static void main(String[] args){
     Student s1 = new Student(111,"Ahmad");
     Student s2 = new Student(222,"Zainab");
     Student.changeUni("QAU");
     s1.display();
     s2.display();

   }
}
```

Important Points





- They can only call other static methods.
- They can only access static data.
- They cannot refer to "this" or "super" (more later) in anyway.

Example

```
1  public class A {
2    int i = 5;
3    static int k = 2;
4
5    public static void main(String[] args) {
6      int j = i; // Wrong because i is an instance variable
7      m1(); // Wrong because m1() is an instance method
8    }
9
10   public void m1() {
11     // Correct since instance and static variables and methods
12     // can be used in an instance method
13     i = i + k + m2(i, k);
14   }
15
16   public static int m2(int i, int j) {
17     return (int)(Math.pow(i, j));
18   }
19 }
```

Example

```
public class Test {
  public int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i ++)
      result *= i;

    return result;
  }
}
```

(a) Wrong design

```
public class Test {
  public static int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++)
      result *= i;

    return result;
  }
}
```

(b) Correct design

Independent of any specific instance.

Example

- Suppose that the class F is defined in (a). Let f be an instance of F. Which of the statements in (b) are correct?

```
public class F {
  int i;
  static String s;

  void imethod() {
  }

  static void smethod() {
  }
}
```

(a)

```
System.out.println(f.i);
System.out.println(f.s);
f.imethod();
f.smethod();
System.out.println(F.i);
System.out.println(F.s);
F.imethod();
F.smethod();
```

(b)

Example – static data and methods

- Create a `SavingsAccount` class.
- Use a static data member `annualInterestRate` to store the annual interest rate.
- The class contains a private data member `savingsBalance` indicating the balance of account.
- Provide member function `calculateMonthlyInterest` that calculates the monthly interest by multiplying the balance by `annualInterestRate` divided by 12; this interest should be added to `savingsBalance`.
- Provide a static member function `modifyInterestRate` that sets the static `annualInterestRate` to a new value.

Example – static data and methods

- Write a driver program to test class `SavingsAccount`. Instantiate two different objects of class `SavingsAccount`, saver1 and saver2, with balances of $2000.00 and $3000.00, respectively.
- Set the `annualInterestRate` to 3 percent.
- Then calculate the monthly interest and print the new balances for each of the savers.
- Then set the `annualInterestRate` to 4 percent, calculate the next month's interest and print the new balances for each of the savers.

Example - Calculator

- Create a class `TwoDigitCalculator` which allows user to perform addition, subtraction, multiplication and division on 2 digits.

- Analyse the data members and methods of this class and implement