



EUROPEAN UNIVERSITY OF BANGLADESH

Assignment : 01
Course name : Object Oriented Programming Sessional
Course code : CSE-212

Submitted to :

Name : Sabrin Afroz
Designation : Lecturer

Submitted by :

Name : MD Sayem Hossen
ID : 250221033
Semister : 2nd
Date : 19 - 11- 2025

1, Static Class

```
class Person {  
  
    // Static method  
    static void sayHello() {  
        System.out.println("Hi");  
    }  
  
    // Non-static method  
    void sayEveryone() {
```

```

        System.out.println("Everyone 😊 ");
    }

}

public class Main {
    public static void main(String[] args) {

        Person.sayHello();
        Person p1 = new Person();

        p1.sayEveryone();
    }
}

```

The screenshot shows a Java code editor interface. On the left, there is a sidebar with icons for different file types: Java (selected), Python, JavaScript, CSS, HTML, XML, JSON, and TypeScript. The main area displays the code for Main.java. The code defines a Person class with a static sayHello method and a non-static sayEveryone method. It also defines a Main class with a main method that calls the static sayHello method, creates a Person object, and calls the non-static sayEveryone method on it. The Run button is highlighted in blue. To the right, the Output window shows the console output: "Hi", "Everyone 😊", and "== Code Execution Successful ==".

```

Main.java

1  class Person {
2
3      // Static method
4      static void sayHello() {
5          System.out.println("Hi");
6      }
7
8      // Non-static method
9      void sayEveryone() {
10         System.out.println("Everyone 😊 ");
11     }
12 }
13
14 public class Main {
15     public static void main(String[] args) {
16
17         Person.sayHello();
18         Person p1 = new Person();
19
20         p1.sayEveryone();
21     }
22 }

```

Output

```

Hi
Everyone 😊
== Code Execution Successful ==

```

2, Static Calculations

```

class Calculate {

    // Method for addition
    void sum(int x, int y) {
        int result = x + y;
        System.out.println("Sum = " + result);

        multiplication(result);
    }

    // Method for multiplication
}

```

```

void multiplication(int a) {
    int m = a * 3;
    System.out.println("Multiplication = " + m);
}

public class Main {
    public static void main(String[] args) {

        Calculate c = new Calculate();

        c.sum(15, 25);
    }
}

```

The screenshot shows a Java code editor interface. On the left, the code for `Main.java` is displayed. The code defines a `Calculate` class with a `sum` method that prints the sum of two integers, and a `multiplication` method that prints the multiplication of an integer by 3. It also defines a `Main` class with a `main` method that creates a `Calculate` object and calls its `sum` method with arguments 15 and 25. On the right, the `Output` pane shows the execution results: `Sum = 40`, `Multiplication = 120`, and `== Code Execution Successful ==`.

```

Main.java
1- class Calculate {
2
3     // Method for addition
4-     void sum(int x, int y) {
5         int result = x + y;
6         System.out.println("Sum = " + result);
7
8         multiplication(result);
9     }
10
11     // Method for multiplication
12-     void multiplication(int a) {
13         int m = a * 3;
14         System.out.println("Multiplication = " + m);
15     }
16 }
17
18- public class Main {
19-     public static void main(String[] args) {
20
21         Calculate c = new Calculate();
22
23         c.sum(15, 25);
24     }
25 }

```

Output

```

Sum = 40
Multiplication = 120
== Code Execution Successful ==

```

3, Static Block

```
class StaticBlock {
```

```
static {
    System.out.println("Static block executed first!");
}
```

```
// Static method
static void calculation() {
    int a = 5;
    int b = 20;
```

```

        int sum = a + b;
        System.out.println("Sum = " + sum);
    }

    public static void main(String[] args) {
        System.out.println("Main method started");
        calculation();
    }
}

```

```

StaticBlock.java | Run | Output | Clear
1~ class StaticBlock {
2
3
4~     static {
5         System.out.println("Static block executed first!");
6     }
7
8     // Static method
9~     static void calculation() {
10        int a = 5;
11        int b = 20;
12        int sum = a + b;
13        System.out.println("Sum = " + sum);
14    }
15
16~    public static void main(String[] args) {
17        System.out.println("Main method started");
18        calculation();
19    }
20 }

```

Static block executed first!
Main method started
Sum = 25
--- Code Execution Successful ---

4, Super Keyword Variable Call

```

class Account {

    int balance = 5000; // parent class variable
}

class SavingsAccount extends Account {

    int balance = 8000; // child class variable

    void showBalance() {
        System.out.println("Child balance = " + balance);
        System.out.println("Parent balance = " + super.balance);
    }
}

```

```

public class Main {
    public static void main(String[] args) {

        SavingsAccount sa = new SavingsAccount();
        sa.showBalance();
    }
}

```

```

Main.java | Run | Output | Clear
1- class Account {|
2
3     int balance = 5000; // parent class variable
4 }
5
6- class SavingsAccount extends Account {
7
8     int balance = 8000; // child class variable
9
10- void showBalance() {
11     System.out.println("Child balance = " + balance);
12     System.out.println("Parent balance = " + super.balance);
13 }
14 }
15
16- public class Main {
17-     public static void main(String[] args) {
18
19         SavingsAccount sa = new SavingsAccount();
20         sa.showBalance();
21     }
22 }

```

Child balance = 8000
Parent balance = 5000
== Code Execution Successful ==

5, Super Keyword Method

```

class Vehicle {

    void speed() {
        System.out.println("Vehicle has a normal speed");
    }
}

```

```

class Bike extends Vehicle {

    void speed() {
        System.out.println("Bike has a high speed");
    }

    void showSpeed() {
        super.speed(); // parent class method call
        speed();      // child class method call
    }
}

```

```

public class Main {
    public static void main(String[] args) {

        Bike b = new Bike();
        b.showSpeed();
    }
}

```

The screenshot shows a Java code editor interface with three tabs: Main.java, Vehicle.java, and Bike.java. The Main.java tab contains the provided code. The Vehicle.java tab contains:

```

1- class Vehicle {
2
3-     void speed() {
4         System.out.println("Vehicle has a normal speed");
5     }
6 }

```

The Bike.java tab contains:

```

8- class Bike extends Vehicle {
9
10-    void speed() {
11        System.out.println("Bike has a high speed");
12    }
13
14-    void showSpeed() {
15        super.speed(); // parent class method call
16        speed();      // child class method call
17    }
18 }

```

The Main.java tab continues with:

```

20- public class Main {
21-     public static void main(String[] args) {
22
23         Bike b = new Bike();
24         b.showSpeed();
25     }
26 }

```

The Output panel shows the console output:

```

Vehicle has a normal speed
Bike has a high speed
== Code Execution Successful ==

```

6, Super Keyword Constructor

```

class Person {

    Person() {
        System.out.println("Person constructor called");
    }
}

class Student extends Person {

    Student() {
        super(); // parent class constructor call
        System.out.println("Student constructor called");
    }
}

public class Main {
    public static void main(String[] args) {

```

```

        Student s = new Student();
    }
}

```

The screenshot shows a Java development environment with the following details:

- Left Sidebar:** Shows icons for various file types: Main.java (selected), Person.java, Student.java, Main.java, JS, TS, and GO.
- Main Area (Main.java):**

```

1~ class Person {
2
3~     Person() {
4         System.out.println("Person constructor called");
5     }
6 }
7
8~ class Student extends Person {
9
10~    Student() {
11        super(); // parent class constructor call
12        System.out.println("Student constructor called");
13    }
14 }
15
16~ public class Main {
17~     public static void main(String[] args) {
18
19         Student s = new Student();
20     }
21 }

```
- Toolbar:** Includes icons for Run, Stop, Share, and Run.
- Output Panel:**

```

Person constructor called
Student constructor called
== Code Execution Successful ==

```

7, Vehicle and Car Programme

```

class Vehicle {
    Vehicle() {
        System.out.println("Vehicle Color");
    }
}

class Car extends Vehicle {
    Car() {
        super();
        System.out.println("BMW car");
    }
}

public class Main {
    public static void main(String[] args) {
        new Car();
    }
}

```



```
Main.java
1+ class Vehicle {
2+     Vehicle() {
3         System.out.println("Vehicle Color");
4     }
5 }
6
7+ class Car extends Vehicle {
8+     Car() {
9         super();
10        System.out.println("BMW car");
11    }
12 }
13
14+ public class Main {
15+     public static void main(String[] args) {
16         new Car();
17     }
18 }
```

Output

```
Vehicle Color
BMW car
==== Code Execution Successful ===
```

– THE END