

Advanced Programming

08/01/2025 | Class - 02

Capstone - CSE498 A → CSE 498 B

↳ Proposal ↳ Coding
↳ Presentation ↳ Defence

loops in Java

↳ "for each" loop

Java has automatic garbage collection

String s[] = { "53 intake", "section 1", ... };

for (String s1: s) {
 System.out.println(s1);
}

array traversal

class names should be unique

forEach()

↳ in JavaScript

OOP Concepts

Class

(blueprint/user defined type/template)

contains variables and methods

↳ attributes
↳ function

also constructors

no memory is allocated when creating class

logical entity

Object

instance of a class

physical entity

memory allocated when an object is created.

* access modifiers will be discussed later

```
class student {  
    private int id;  
    private String name;  
    private double cgpa;
```

```
    void set(int id, String name, double cgpa) {
```

```
        this.id = id;
```

```
        this.name = name;
```

```
        this.cgpa = cgpa;
```

→ key word **

```
    }
```

```
    void get() {
```

```
        System.out.println (
```

```
            "id: " + id
```

```
            + "name: " + name
```

```
            + "cgpa: " + cgpa
```

```
        );
```

```
    }
```

```
public class lab1 {
```

```
    public static void main (String[] args) {
```

```
        Student s1 = new Student();
```

data type variable

new instance

```
        s1.set(408, "Shayan", 3.72);
```

```
        s1.show();
```

```
    }
```

```
}
```

When no constructor is declared, default constructor is called

Object Creation
↳ Dynamic Memory Allocation


```
class Box {
```

```
    private double height;
```

```
    private double width;
```

```
    private double depth;
```

```
    void set(double height, double width, double depth) {
```

```
        this.height = height;
```

```
        this.width = width;
```

```
        this.depth = depth;
```

```
    }  
    void volume() {
```

```
        System.out.println
```

```
        ("Volume: " + (height * width * depth));  
    }
```

```
}  
}  
public class cl1 {
```

```
    public static void main(String[] args) {
```

```
        Box B1 = new Box();
```

```
        Box B2 = new Box();
```

```
        B1.set(10, 20, 30);
```

```
        B1.volume();
```

```
        B2.set(40, 50, 60);
```

```
        B2.volume();  
    }  
}
```

```
}
```

constructor

- ↳ a method that initializes a class

constructor

- ↳ default
- ↳ parameterized
- ↳

constructor name = class name

<overloading concept>

```
class Student{
```

```
    private int id;
```

```
    private String name;
```

```
    private double cgpa;
```

```
    Student(int id, String name, double cgpa){  
        ...  
    }
```

```
    Student(){  
        ...  
    }
```

```
}
```

Type signature is used to identify which constructor should be used in constructor overloading.


```
class student {  
    ...  
    Student (Student s3) {  
        this.id = s3.id;  
    }  
}
```

expects an object

```
class Box {  
    private double height, width, depth;  
    Box() {  
        this.height = 10;  
        this.width = 20;  
        this.depth = 30;  
    }  
    Box(double height, double width, double depth) {  
        this.height = height;  
        this.width = width;  
        this.depth = depth;  
    }  
    void show() {  
        System.out.println(  
            "Volume: " + (height * width * depth)  
        );  
    }  
}
```

```

public class cl2 {
    public static void main(String[] args) {
        Box B1 = new Box(30, 60, 90);
        Box B2 = new Box();
        B1.show();
        B2.show();
    }
}

```

Next class : Inheritance

Mid : <AU code>

Theory : Max 2-3 marks