ÖZYEĞİN UNIVERSITY

CS 102

Object Oriented Programming

Static Class Members

Reyyan Yeniterzi

reyyan.yeniterzi@ozyegin.edu.tr

Static members

- In defalt, each object has its own class members.
- Class members are not shared among object.

Static members

- However, in certain cases a class member (variable or method) needs to be shared among all objects of the class.
- For example:
  - Counting the number of insances of the class
  - One can use this number as unique id of the object
  - At the same time the total count will be hold at the class

Static members

- Such class variables are defined as static by using the keyword `static`
- These variables and methods belong to the class, not to any instance of the class.

Static members

- Such class variables are defined as static by using the keyword `static`
- These variables and methods belong to the class, not to any instance (object) of the class.
- Static vs. Non-static
  - Static: class variable/method
  - Non-static: instance variable/method

Static members

- A static class member is shared by all objects of the class.
- They all access to the same data.

Example


```
public class Student {
    private String name;
    private int id;
    private static int studentCount = 0;

    public Student(String name) {
        this.name = name;
        this.id = ++studentCount;
    }
    public int getID() {
        return id;
    }
    public String getName() {
        return name;
    }
    public static int getStudentCount() {
        return studentCount;
    }
}
```

Example


```
public class Student {
    private String name;
    private int id;
    private static int studentCount = 0;

    public Student(String name) {
        this.name = name;
        this.id = ++studentCount;
    }
    public int getID() {
        return id;
    }
    public String getName() {
        return name;
    }
    public static int getStudentCount()
        return studentCount;
    }
}
```

- Static keyword can be used for both
  - Class instances (variables)
  - Class methods

Static vs. Non-Static

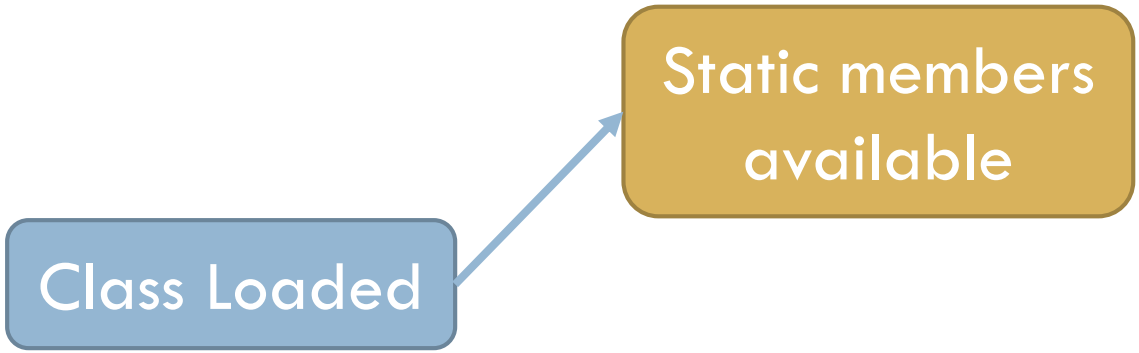
- Non-static variables need an object to be initialized in order to access them.
- Static class variables can be initialized without instantiating an object of the class.

Accessing Static Members

Class Loaded

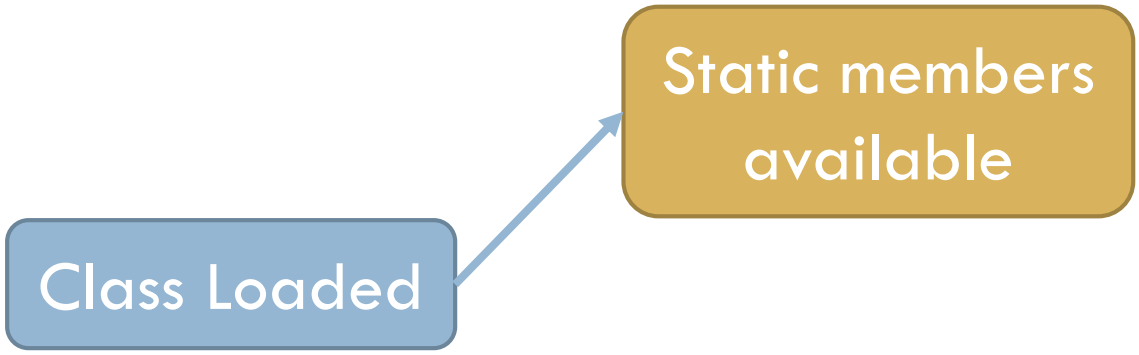
- Class is loaded by JVM

Accessing Static Members



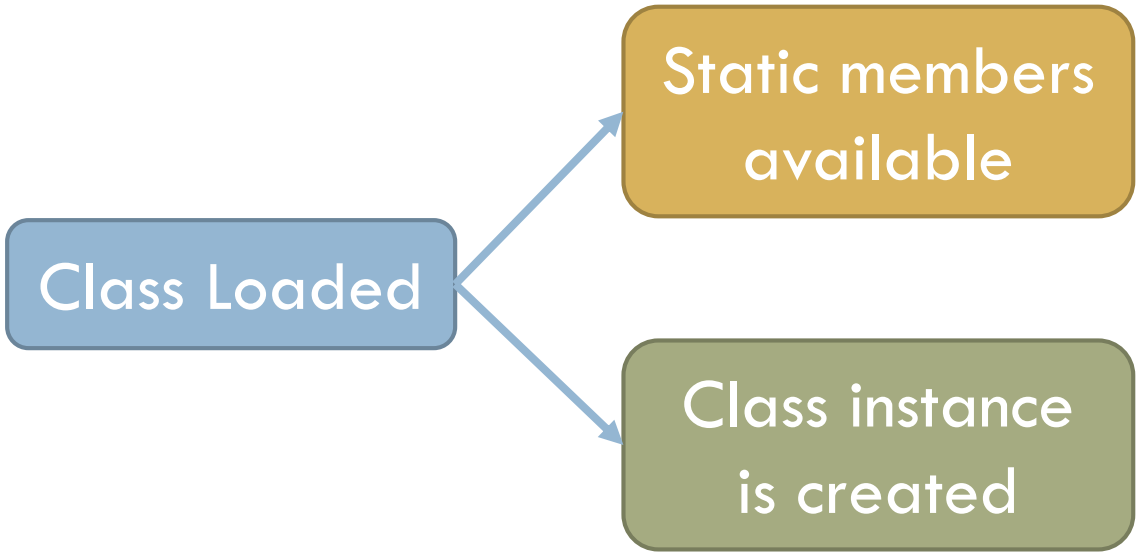
- Static variable and methods are loaded and initialized.
- They are ready to use.

Accessing Static Members



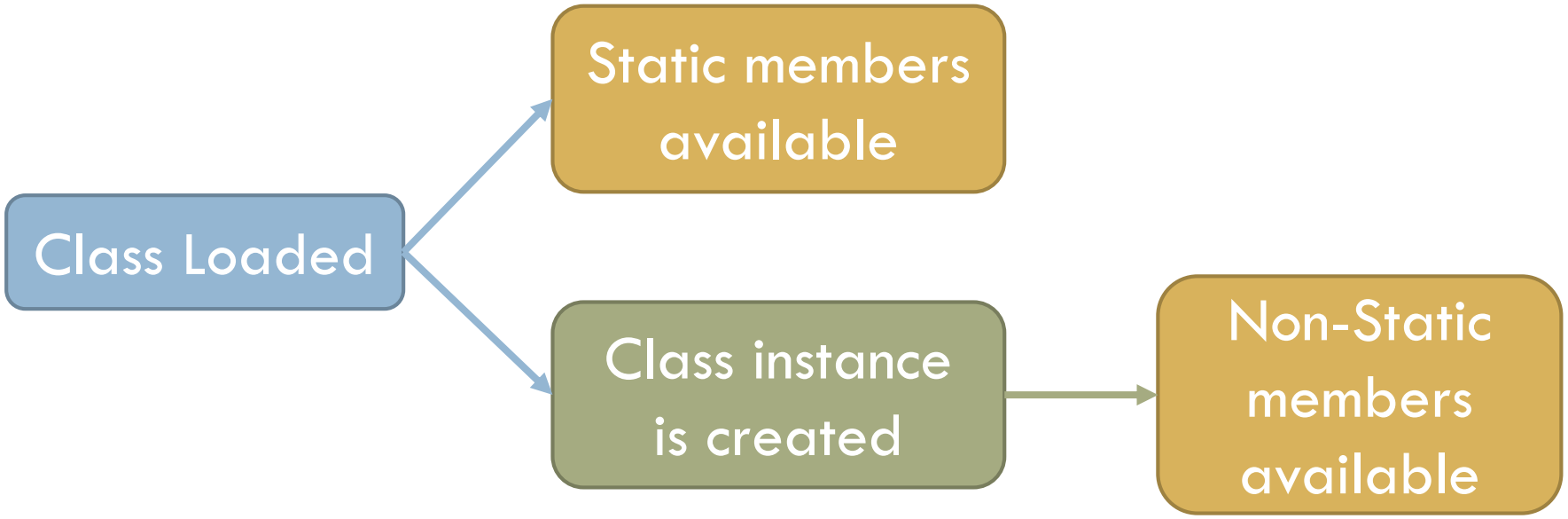
- As seen, we don't need to create any class instance to access them.

Accessing Static Members



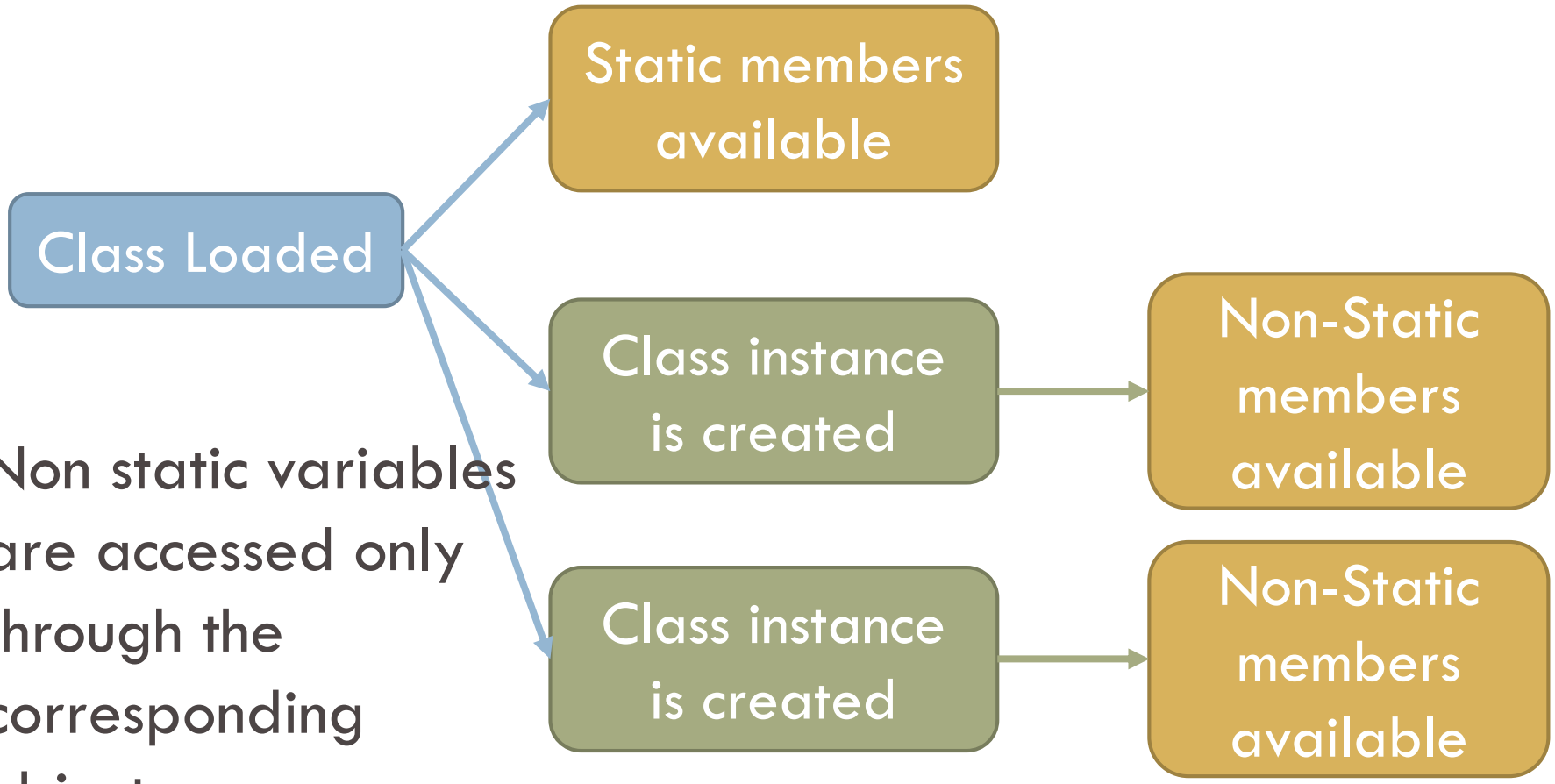
- Objects are instantiated by calling the constructor.

Accessing Static Members



- Non static variables and methods are created.
- They are ready to use.

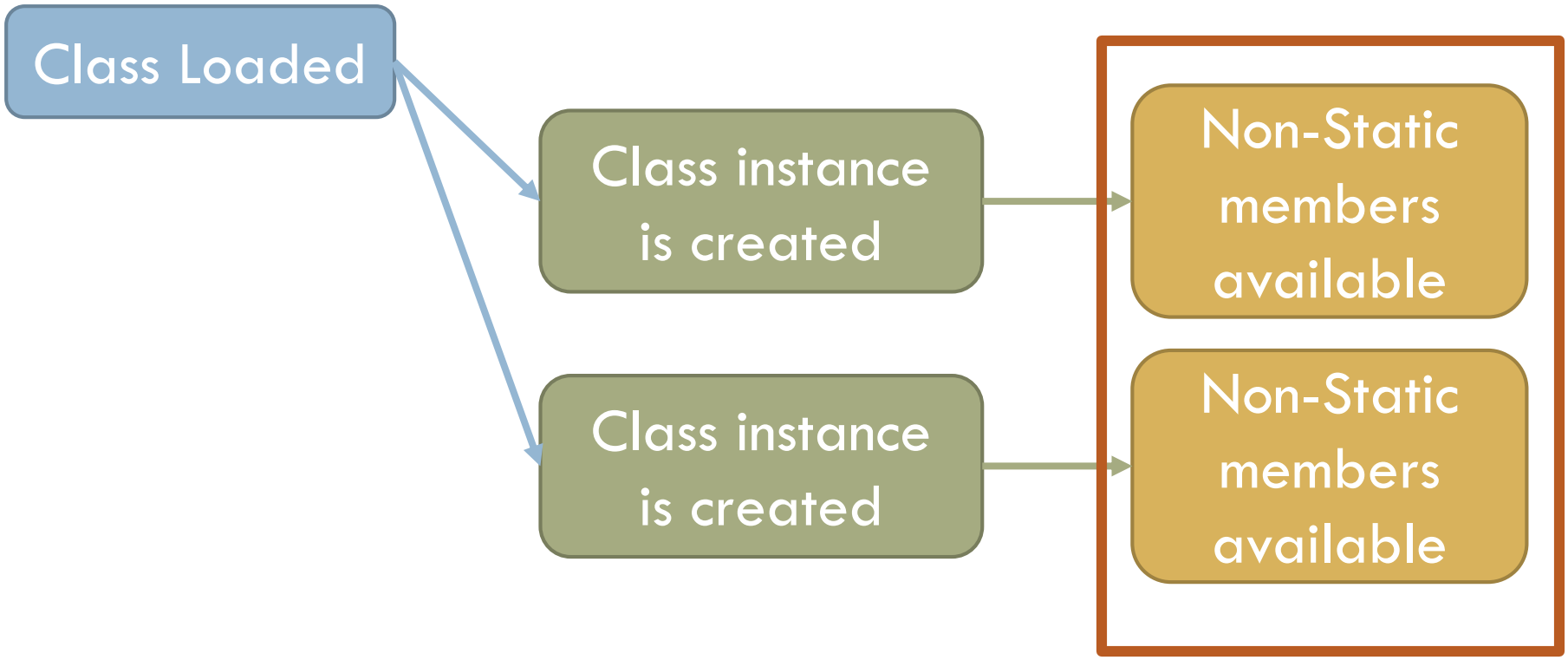
Accessing Static Members



- Non static variables are accessed only through the corresponding objects.

Accessing Static Members

- The values of non-static fields make one object distinct from another



Static members

- Static keyword can be used for both
  - Class instances (variables)
  - Class methods

Static class variables

- If static variable is not initialized, the compiler assigns a default value.

Static class variables

- If static variable is not initialized, the compiler assigns a default value.
- If it is not private, it can be accessed like

```
ClassName.variableName
```

Accessing Static Class Variable


```
public class Student {
    private String name;
    private int id;
    protected static int studentCount = 0;

    public Student(String name) {
        this.name = name;
        this.id = ++studentCount;
    }
    public static int getStudentCount() {
        return studentCount;
    }
}

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

        System.out.println(Student.studentCount);
    }
}
```

Accessing Static Class Variable


```
public class Student {
    private String name;
    private int id;
    protected static int studentCount = 0;

    public Student(String name) {
        this.name = name;
        this.id = ++studentCount;
    }
    public static int getStudentCount() {
        return studentCount;
    }
}

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

        System.out.println(Student.studentCount);
    }
}
```

```
0
```

Accessing Static Class Variable


```
public class Student {
    private String name;
    private int id;
    protected static int studentCount = 0;

    public Student(String name) {
        this.name = name;
        this.id = ++studentCount;
    }
    public static int getStudentCount() {
        return studentCount;
    }
}

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

        System.out.println(Student.studentCount);
        Student s = new Student("Ali");

        System.out.println(Student.studentCount);
    }
}
```

Accessing Static Class Variable


```
public class Student {
    private String name;
    private int id;
    protected static int studentCount = 0;

    public Student(String name) {
        this.name = name;
        this.id = ++studentCount;
    }
    public static int getStudentCount() {
        return studentCount;
    }
}

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

        System.out.println(Student.studentCount);
        Student s = new Student("Ali");

        System.out.println(Student.studentCount);
    }
}
```

```
0
1
```

Static Class Methods

- Static methods cannot access non-static class variables, why?

Static Class Methods

- Static methods cannot access non-static class variables, why?
  - These methods can be called even if there are not any object of the class have been instantiated.
  - Since there is no object. `this` keyword, the calling object, cannot be used in static methods.
  - It needs to reference to a specific object. When this method is called , there may not be any objects.

Static Class Methods

- Static methods cannot call non-static methods.
- In case user does, we will get compiler error.

Static Class Methods

- Static methods cannot call non-static methods.
- In case user does, we will get compiler error.

- Non-static methods can access
  - All static variables and methods
  - All non-static variables and methods

Static Class Methods

- Static methods, can be invoked with the class name, without the need for creating an instance of the class

```
ClassName.methodName(args)
```

Static Class Methods


```
public class Student {
    private String name;
    private int id;
    private static int studentCount = 0;

    public Student(String name) {
        this.name = name;
        this.id = ++studentCount;
    }
    public static int getStudentCount() {
        return studentCount;
    }
}

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

        System.out.println(Student.getStudentCount());

        Student s = new Student("Ali");

        System.out.println(s.getStudentCount());
        System.out.println(Student.getStudentCount());
    }
}
```

Static Class Methods


```
public class Student {
    private String name;
    private int id;
    private static int studentCount = 0;

    public Student(String name) {
        this.name = name;
        this.id = ++studentCount;
    }
    public static int getStudentCount() {
        return studentCount;
    }
}

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

        System.out.println(Student.getStudentCount());

        Student s = new Student("Ali");

        System.out.println(s.getStudentCount());
        System.out.println(Student.getStudentCount());
    }
}
```

```
0
1
1
```

Static Class Methods

- *main()* method is static
  - It must be accessible for an application to run, before any instantiation takes place.

Summary

- Static members are useful for sharing data accross class objects
- Static variables
  - They belong to the class, not to a class instance
  - They are initialized as the program starts running, before instantianitaion of any class object
  - They an be accessed directly by using class name, they don't need an object.

Summary

- Static methods
  - They belong to the class, not an object.
  - They can only access static methods and variables
  - They cannot access instance variables and methods. They cannot use `this` or `super` keywords, since there is instance to refer.
  - They can be called by using the class name.

Any Questions ?