WEEK 8

User-Defined Class (Static Class Members and Method Invocation)

Sharmila Mat Yusof

Outline

- Static Variable and Method
  - Example of the use of static variable and method in class
- Objects as a Parameter(s) and Return Type
  - Difference between primitive-type arguments and object-type arguments
  - Example of passing objects to method
- Array of Objects
  - Develop methods with object argument(s)
  - Store and process objects in array

Learning Objectives

- To differentiate between static variable and method
- To differentiate between primitive-type arguments and object-type arguments
- To demonstrate a program that pass objects to method
- To develop methods with object arguments
- To store and process objects in array

Static Variables

- Also known as class variables.
- Only one copy of static variable (single storage location) is created for a class.
- Use `static` keyword to declare a static variable.
- Static constants are often declared as public.
- Syntax:

```
<accessModifier> static <dataType> <variableName>;
```

- Example:

```
public static int noOfStudent;
```

Static Methods

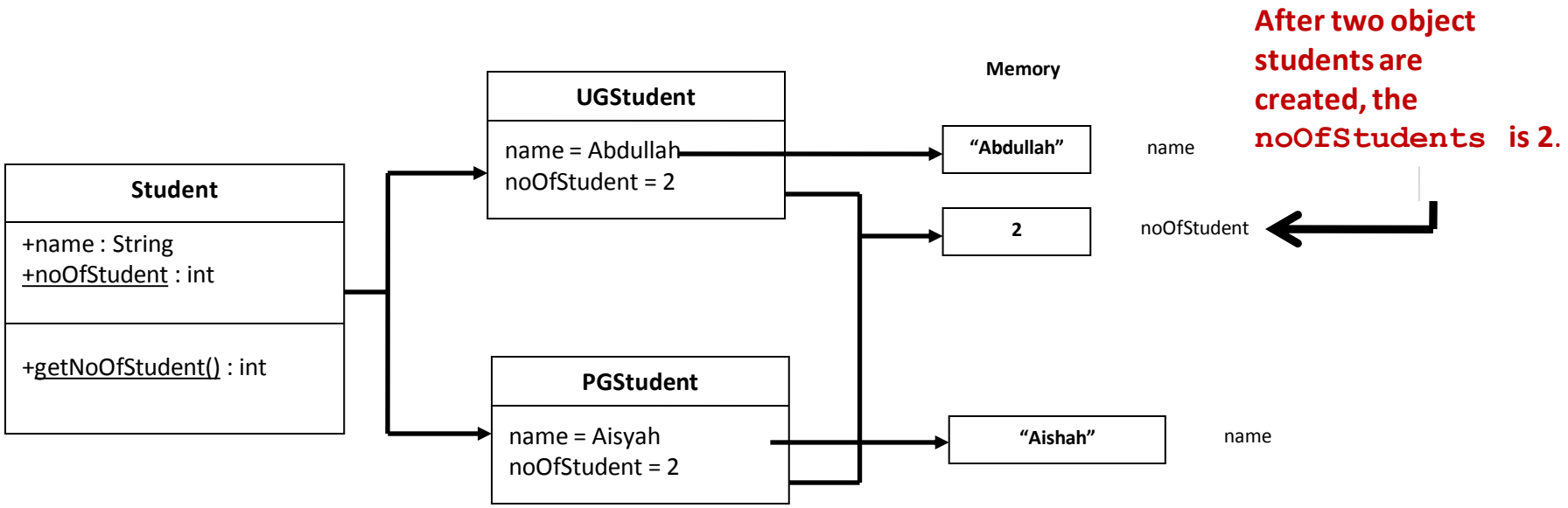
- Also known as class methods.
- Normally defined to access and change static variables.
- Associated with the class, not with any object.
- Cannot access instance variables.
- Can be called before any object is instantiated.

Rules for Static and Non-Static Methods

	Static Method	Non-static Method
Access instance variables?	no	yes
Access static variables?	yes	yes
Invoke static methods?	yes	yes
Invoke non-static instance methods?	no	yes
Use the object reference this?	no	yes

Static Variables and Methods: Example

- The following diagram illustrates the roles of instance and class variables and their uses. This example adds a class variable **`noOfStudent`** to track the number of Student objects created.



Static Variables and Methods: Example

Student class

```
1   package student;
2
3   public class Student
4   {
5       String name;
6       char grade;
7       public static int noOfStudent = 0;
8
9
10      public Student(String nName)
11      {
12          name = nName;
13          noOfStudent++;
14      }
15
16      public String getName( )
17      {
18        return name;
19      }
20
21      public static int getNoOfStudent()
22      {
23        return noOfStudent;
24      }
25  }
```

static variable

static method

Static Variables and Methods: Example

Client class

Accessing a static variable

```
1   package student;
2   public class testStaticMembers
3   {
4     public static void main(String[] args)
5     {
6         Student UGStudent = new Student("Abdullah");
7         System.out.println("Student's name: "+UGStudent.name);
8         System.out.println("Number of Student: "+UGStudent.noOfStudent);
9         Student PGStudent = new Student("Aisyah");
10        System.out.println("Student's name: "+PGStudent.name);
11        System.out.println("Number of Student: "+PGStudent.noOfStudent);
12        UGStudent = null;
13        PGStudent = null;
14        System.out.println("Number of Student after objects are deleted: "+Student.getNoOfStudent());
15    }
16  }
```

Calling a static or class method

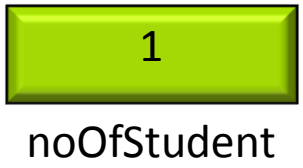
Static Variables and Methods: Example

Output:

```
run:
Student's name: Abdullah
Number of Student: 1
Student's name: Aisyah
Number of Student: 2
Number of Student after objects are deleted: 2
BUILD SUCCESSFUL (total time: 1 second)
```

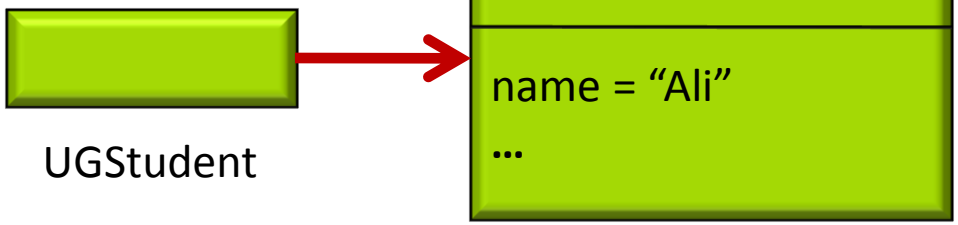
Primitive Data Types vs. Object Types Variables

Primitive type int noOfStudent = 1;



Created using ***new*** Student()

Object type Student UGStudent;



Copying Variable of Primitive Data Types and Object Types

Primitive type assignment:

a = b;

Before:

a 2

b 4

After:

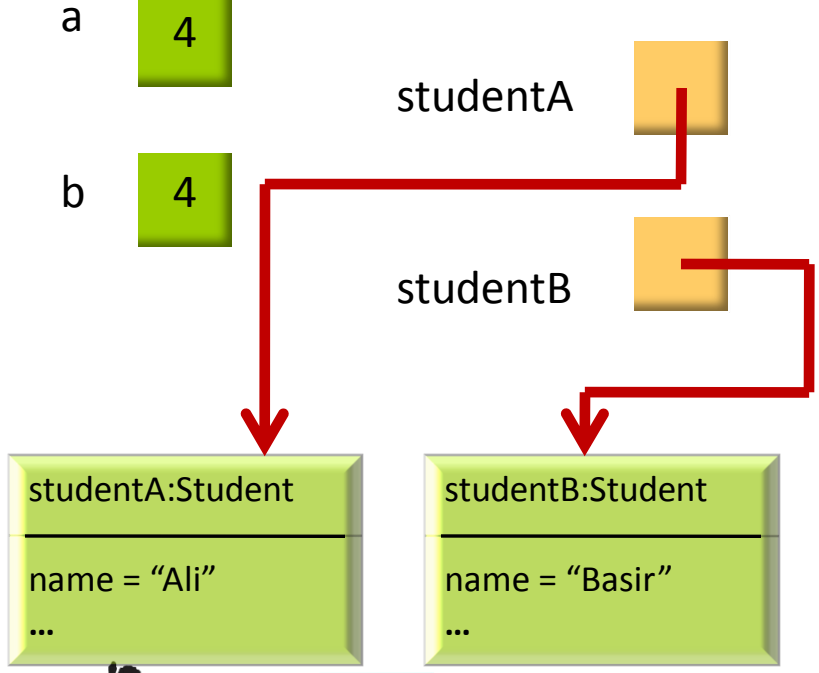
a 4

b 4

Object type assignment:

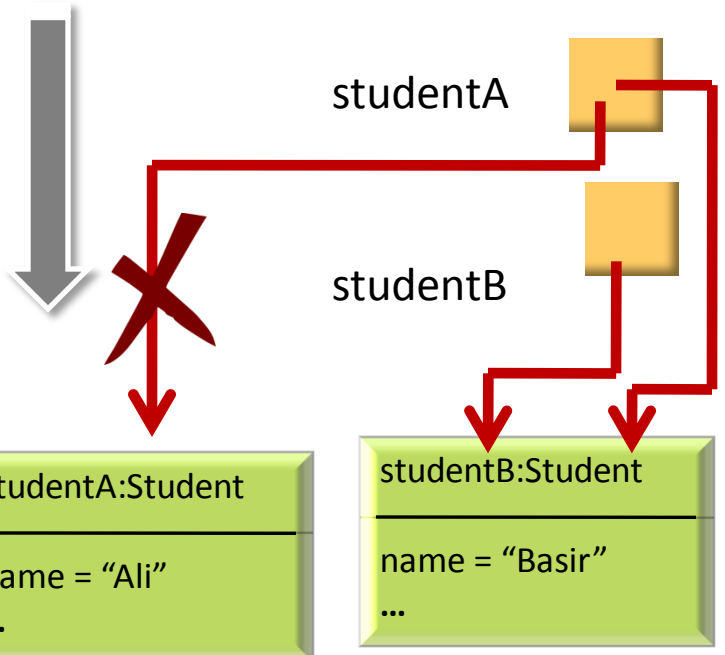
studentA = studentB;

Before:



***Note**: JVM automatically collect the garbage.*

After:



Passing Object to Method

- Two ways to pass argument(s) to methods:
  - Passing by value for primitive type variable - the value is passed to the parameter.
  - Passing by value for reference type variable -the value is the reference to the object.
- Reference type variable allows an object to be referred multiple times.
- So far, we have seen the passing by value for primitive type variable. We will next demonstrate passing by value for reference type variable.

Passing Object to Method: Example

Student class[1]

```
1   package student;
2   public class Student {
3       String name;
4       int matricNo;
5       public String grade;
6       public static int noOfStudent = 0;
7
8       public Student(String studName,int matricNum,double mark)
9        {
10         name = studName;
11         matricNo = matricNum;
           grade = determineGrade(mark);
13         noOfStudent++;
14       }
15      public String getName()
16      {
17          return name;
18      }
19        public int getMatricNo()
20      {
21          return matricNo;
22      }
```

Passing Object to Method: Example

Student class[2]

```
23      public String determineGrade(double mark)
24      {
25       if (mark > 39)
26           grade = "PASS";
27       else
28           grade = "FAIL";
29       return grade;
30      }
31      public void displayInfo()
32      {
33        System.out.println("Name: "+name);
34        System.out.println("Matric Number: "+matricNo);
35        System.out.println("Grade: "+grade);
36      }
37
38      public void displayInfo(Student stud,double sMark)
39       {
40           stud.name="Sabila";
41           stud.matricNo = 33333;
42           stud.determineGrade(sMark);
43       }
44  }
```

the value of studMark is passed to sMark

UGStudent & stud are pointing to the same object Student

displayInfo() method is invoked

Passing Object to Method: Example

Client class

```
1   package student;
2   import java.util.*;
3   public class PassObject
4   {
5     public static void main(String[] args){
6
7         Scanner read = new Scanner(System.in);
8         double studMark;
9
10        Student UGStudent = new Student("Adila",22222,30);
11        UGStudent.displayInfo();
12        System.out.println();
13
14        System.out.print("Please enter student's mark: " );
15        studMark = read.nextDouble();
16
17        UGStudent.displayInfo(UGStudent,studMark);
18        System.out.println();
19        System.out.println("After object passed to method ");
20        System.out.println("Student's name :"+UGStudent.getName());
21        System.out.println("Student's matricNo :"+UGStudent.getMatricNo());
22        System.out.println("Student's grade :"+UGStudent.determineGrade(studMark));
23        }
24    }
```

Pass by reference (UGStudent)

Pass by value (studMark)

Invoke displayInfo() method

Passing Object to Method: Example

Output:

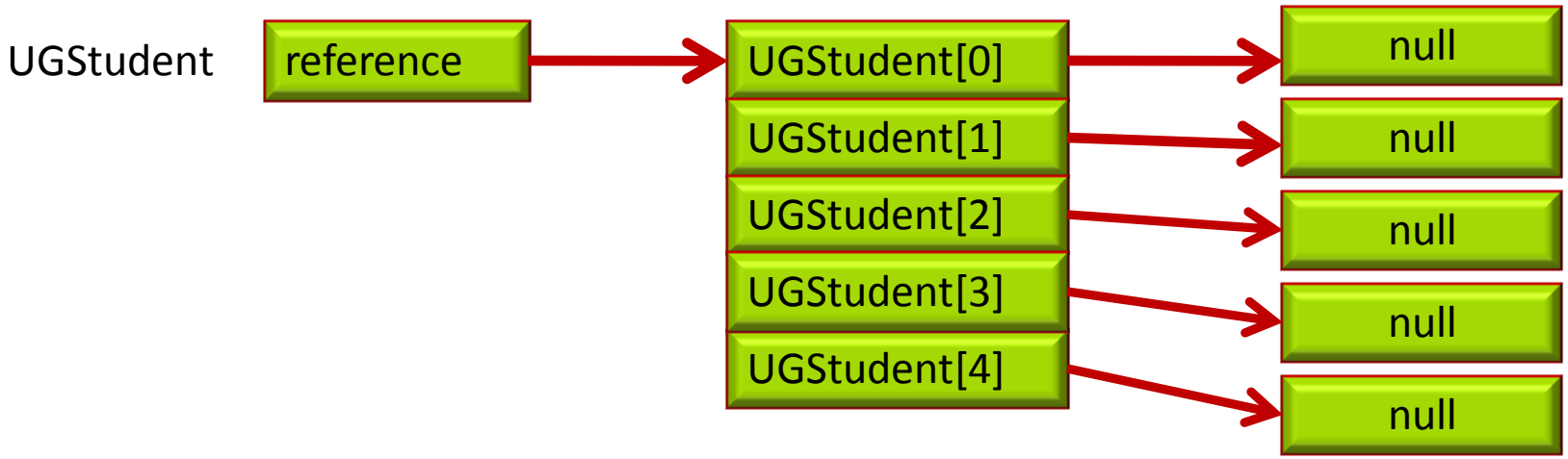
```
run:
Name: Adila
Matric Number: 22222
Grade: FAIL

Please enter student's mark: 85

After object passed to method:
Student's name :Salsabila
Student's matricNo :33333
Student's grade :PASS
BUILD SUCCESSFUL (total time: 6 seconds)
```

Array of Objects

- In Java, array elements are not limited to primitive data types (e.g `int`, `double`, `float`, `char` etc.), we can also has objects as an element of array.
- An array of objects can be more powerful than primitive data types.
- The use of an array of objects allows us to present our application in more simple and rational way.

Array of Objects

```
Student[] UGStudent = new Student[5];
```



Array of Objects

- UGStudent references to the entire array.
- UGStudent[index] references to a Student object.
- Initially an array of Student objects holds `null` references.
- Each object stored in an array must be instantiated separately.

Array of Objects: Example

Consider the following Java code to create five **`Student`** objects:

```
package student;

public class createObjects
{
  public static void main(String[] args)
  {
       Student Stud1 = new Student("Aziz");
       Student Stud2 = new Student("Basir");
       Student Stud3 = new Student("Charles");
       Student Stud4 = new Student("Darwin");
       Student Stud5 = new Student("Elis");
  }
}
```

Array of Objects: Example

The previous program can also be written using an array as follows:

```
1   package student;
2   import java.util.*;
3   public class objectArray
4   {
5     public static void main(String[] args)
6     {
7         Scanner read = new Scanner(System.in);
8         int numStud = 5;
9         Student[] UGStudent = new Student[numStud];
10
11        for(int index=0;index < numStud;index++)
12         {
13           System.out.print("Please input the name of student "+(index+1)+": ");
14           String studName = read.nextLine();
15           UGStudent[index] = new Student(studName);
16         }
17      }
18  }
```

The array is created using **`Student`** object.

Array of Objects: Example

Objects can be handled using loops (i.e `for` or `while` loop) if an array of objects is used in a program.

```
17          System.out.println("The name of student are:  ");
18           for(int index=0;index < numStud;index++)
19           {
20
21             System.out.print("Student "+(index+1)+ ":  ");
22             System.out.println(UGStudent[index].name);
23           }
             System.out.println("Total student is "+UGStudent[numStud-1].noOfStudent);
25       }
26   }
```

Array of Objects: Example

Output:

```
run:
Please input the name of student 1: Aziz
Please input the name of student 2: Basir
Please input the name of student 3: Charles
Please input the name of student 4: Darwin
Please input the name of student 5: Elis
The name of student are:
Student 1:  Aziz
Student 2:  Basir
Student 3:  Charles
Student 4:  Darwin
Student 5:  Elis
Total student is 5
BUILD SUCCESSFUL (total time: 52 seconds)
```

Summary

- Static variable or class variable will only have single storage location for a class
- Static method or class method is not associated with any object. Thus, the method can be called without any object creation.
- Two ways to pass arguments to methods:
  - Passing by value for primitive type variable where the value is passed to the parameter.
  - Passing by value for reference type variable where the value is the reference to the object.
- Other than primitive data types such as `int`, `char` etc., an object can also become the element of an array.
- The use of an array of objects allows us to present our application in more simple and rational way.