Week 6

User-Defined Class

(Object Instantiation)

Norida Muhd Darus

Outline

- Creating Object
  - Object reference variable declaration syntax
  - Object instantiation syntax
  - Constructor method
  - Examples of object instantiation using default constructor and common constructor
  - The Object Reference *this*
- Accessing Class Members
  - Accessing class members syntax
  - Example of accessing class members

Learning Objectives

- To demonstrate how to create object from a user-defined class
- To initialize objects using constructors
- To demonstrate using the object reference ***this***
- To access an object's data and methods using the object member access dot operator (.)

Creating Object

- To use user-defined classes in a program, we need to create an object from the user-defined class.
- Two steps to create an object:
  - STEP1
    - Declaration – declare object reference variable
  - STEP 2
    - Object instantiation – create an object

Creating Object:
Step 1 Declaration Syntax

- Syntax to declare object reference variable:

  <ObjectType> <objectName>;

  where:

  - ***<ObjectType>*** – The type of object (a class name).
  - ***<objectName>*** – the name of object reference variable.

- *objectName* will store the location of the object *ObjectType* in the memory.

Creating Object: Step 1 Examples

- Examples:

```
public class ObjectInstantiation {
    public static void main (String [] args){
        Student UGStudent;//declare object reference variable
    }
}
```

1

Creating Object: Step 2 Instantiation

Syntax

- Syntax to instantiate an object:

<objectName> = new <ObjectType>(<parameter(s)>);

where:

- ***<objectName>***– name of object reference variable.
- ***<ObjectType> (<parameter(s)>)*** – constructor to set up the object.

- Examples:

```
public class ObjectInstantiation {
    public static void main (String [] args){
        Student UGStudent;//declare object reference variable
        UGStudent= new Student("Ahmad", 112233);//create the object
    }
}
```

Creating Object:
Step 1 and Step 2 in one statement

- Syntax:

```
<ObjectType> <objectName> = new <ObjectType>(<parameter(s)>);
```

- Examples:

```
public class ObjectInstantiation {
    public static void main (String [] args){
        Student UGStudent = new Student("Ahmad", 112233);//declare and create the object
    }
}
```

Constructor Method

- Special methods that are invoked to construct objects using the **new** keyword.
- Constructors must have the same name as the class name.
- Constructors play the role of object initialization.
- A class can have several constructors.

Default Constructor Method

- A class may be defined without constructors.
- In this case, a no-argument constructor with an empty body is implicitly declared in the class.
- This constructor, called a **default constructor**, is provided automatically only if no constructors are explicitly defined in the class.

Constructor Method: Method Definition Syntax

- Syntax to define a constructor method:

```
public <ClassName> ( <parameter(s)>)
{
        // constructor body
}
```

- Constructor method definition is the same with other member methods.
- Each constructor must have a different number of parameters or parameters of different types
- However, constructor method has no return value, not even `void`!

Constructor Method: Default Initial Values

- If the constructor does not assign any value to the instance variables, the following default values shall be automatically assigned:

Data Type	Default Value
byte, short, int, long	0
float, double	0.0
char	space
boolean	false
Any object reference (e.g. String)	null

Default Constructor Method: Example

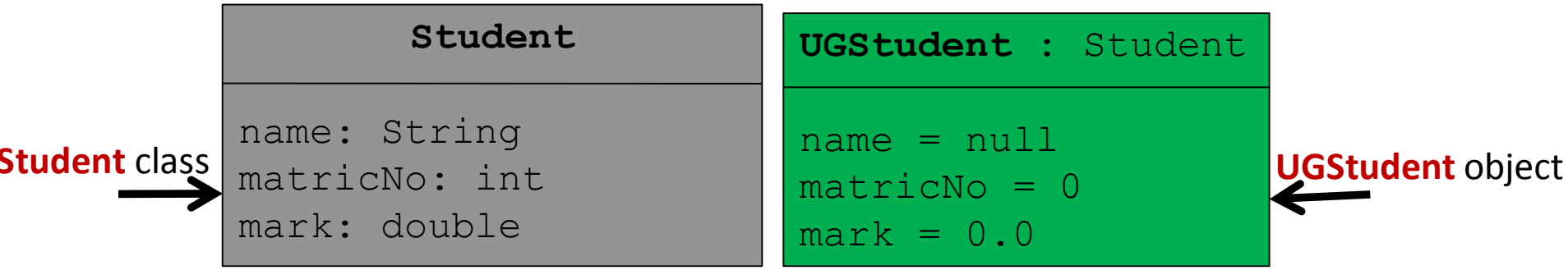
```
public class Student {
    //variable
    String name;
    int matricNo;
    double mark;

    //method
    public String determineGrade(){
        String grade;
        if (mark > 39)
            grade="PASS";
        else
            grade="FAIL";
        return grade;
    }

    public void displayInfo(){
        System.out.println("Your name: "+name);
        System.out.println("Your matric number: "+matricNo);
        System.out.println("Your mark: "+mark);
        System.out.println("Your grade: "+determineGrade());
    }
}
```

- There is no constructor methods defined in the above **`Student`** class.
- Thus, a **no-argument constructor** with an empty body is implicitly declared in the class.

Object Instantiation Using Default Constructor

```
public class ObjectInstantiation {
    public static void main (String [] args){
        Student UGStudent = new Student();//declare and create the object
        UGStudent.displayInfo();
    }
}
```

The state of the UGStudent object after execution of the above codes:



Object Instantiation Using Default Constructor

- Output:

```
run:
Your name: null
Your matric number: 0
Your mark: 0.0
Your grade: FAIL
BUILD SUCCESSFUL (total time: 1 second)
```

Constructor Method: Example

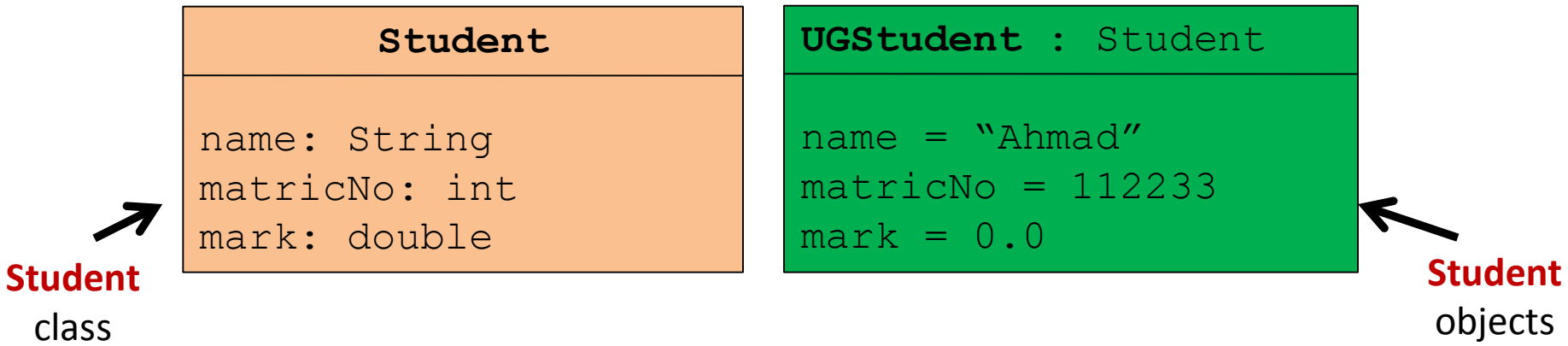
```
public class Student {
    //variable
    String name;
    int matricNo;
    double mark;

    //method
    public Student(String studName, int studMatric){//constructor
        name=studName;
        matricNo=studMatric;
    }
    public Student(String name, int matricNo, double mark){//constructor
        this.name=name;
        this.matricNo=matricNo;
        this.mark=mark;
    }
    public String determineGrade(){
        String grade;
        if (mark > 39)
            grade="PASS";
        else
            grade="FAIL";
        return grade;
    }
```

- The above **`Student`** class have two constructor methods with different parameters.

Object Instantiation Using Constructor

```
public class ObjectInstantiation {
    public static void main (String [] args){
        Student UGStudent = new Student("Ahmad", 112233); //declare and create the object
        UGStudent.displayInfo();
    }
}
```

The state of the UGStudent object after execution of the above codes:



Object Instantiation Using Constructor

- Output:

```
run:
Your name: Ahmad
Your matric number: 112233
Your mark: 0.0
Your grade: FAIL
BUILD SUCCESSFUL (total time: 3 seconds)
```

The Object Reference *this*

- Consider the following Java code fragment:

```
7       //method
8       public Student(String studName, int studMatric){//constructor
9           name=studName;
10          matricNo=studMatric;
11      }
12      public Student(String name, int matricNo, double mark){//constructor
            name=name;
            matricNo=matricNo;
            mark=mark;
16      }
```

- The above codes use the same name for instance variables and parameters which resulted in compiler treating the variables inside the method as parameters.

The Object Reference *this*

- There are two common solutions to the previous problem:
  - Use the different name for the instance variable and parameter.
  - Use ***this*** to refer to an instance variable
- ***this*** is an *implicit parameter* sent to methods and is an object reference to the object for which the method was called.

The Object Reference *this*: Solution 1

```
public class Student {
    //variable
    String name;
    int matricNo;
    double mark;

    //method
    public Student(String studName, int studMatric){//constructor
        name=studName;
        matricNo=studMatric;
    }
    public Student(String name, int matricNo, double mark){//constructor
        this.name=name;
        this.matricNo=matricNo;
        this.mark=mark;
    }
    public String determineGrade(){
        String grade;
        if (mark > 39)
            grade="PASS";
        else
            grade="FAIL";
        return grade;
    }
```

Use different name

- *name* and *matricNo* refer to the instance variables.
- *studName* and *studMatric* refer to the parameters.

The Object Reference *this*: Solution 2

```
public class Student {
    //variable
    String name;
    int matricNo;
    double mark;

    //method
    public Student(String studName, int studMatric){//constructor
        name=studName;
        matricNo=studMatric;
    }
    public Student(String name, int matricNo, double mark){//constructor
        this.name=name;
        this.matricNo=matricNo;
        this.mark=mark;
    }
    public String determineGrade(){
        String grade;
        if (mark > 39)
            grade="PASS";
        else
            grade="FAIL";
        return grade;
    }
```

Use object reference **this**

- *this.name, this.matricNo* and *this.mark* refer to the instance variables.
- *name, matricNo* and *mark* refer to the parameters.

Accessing Class Members

- Once object is created, we can use the **dot operator (.)** to accessed its data or invoke its methods.
- Syntax to access data:

```
<objectName>.<instanceVariable>;
```

- Syntax to access or invoke method:

```
<objectName>.<methodName(argument(s))>;
```

Accessing Class Members: Examples

```
public class Student {
    //variable
    String name;
    int matricNo;
    double mark;

    //method
    public Student(){//default constructor
    }

    public Student(String name, int matricNo){//constructor
        this.name=name;
        this.matricNo=matricNo;
    }

    public String determineGrade(){
        String grade;
        if (mark > 39)
            grade="PASS";
        else
            grade="FAIL";
```

Accessing data

```
public class ObjectInstantiation {
    public static void main (String [] args){
        Student UGStudent = new Student("Ahmad", 112233); //declare and create the object
        UGStudent.mark=85.5;
        UGStudent.displayInfo();
    }
}
```

Accessing Class Members: Examples

```
    public String determineGrade(){
        String grade;
        if (mark > 39)
            grade="PASS";
        else
            grade="FAIL";
        return grade;
    }

    public void displayInfo(){
        System.out.println("Your name: "+name);
        System.out.println("Your matric number: "+matricNo);
        System.out.println("Your mark: "+mark);
        System.out.println("Your grade: "+determineGrade());
    }
}
```

Invoking method

```
public class ObjectInstantiation {
    public static void main (String [] args){
        Student UGStudent = new Student("Ahmad", 112233); //declare and create the object
        UGStudent.mark=85.5;
        UGStudent.displayInfo();
    }
}
```

Accessing Class Members : Examples

- Output:

```
run:
Your name: Ahmad
Your matric number: 112233
Your mark: 85.5
Your grade: PASS
BUILD SUCCESSFUL (total time: 2 seconds)
```

Summary

- There are two steps in creating object:
  - Step 1: Declare object reference variable
  - Step 2: Instantiate object
- Object is initialized using constructor/default constructor.
- The object reference ***this*** can be used to differentiate between instance variables and parameter variables.
- A created object use the **dot operator (.)** to accessed its data or invoke its methods.