WEEK 3

Classes and Objects in Java

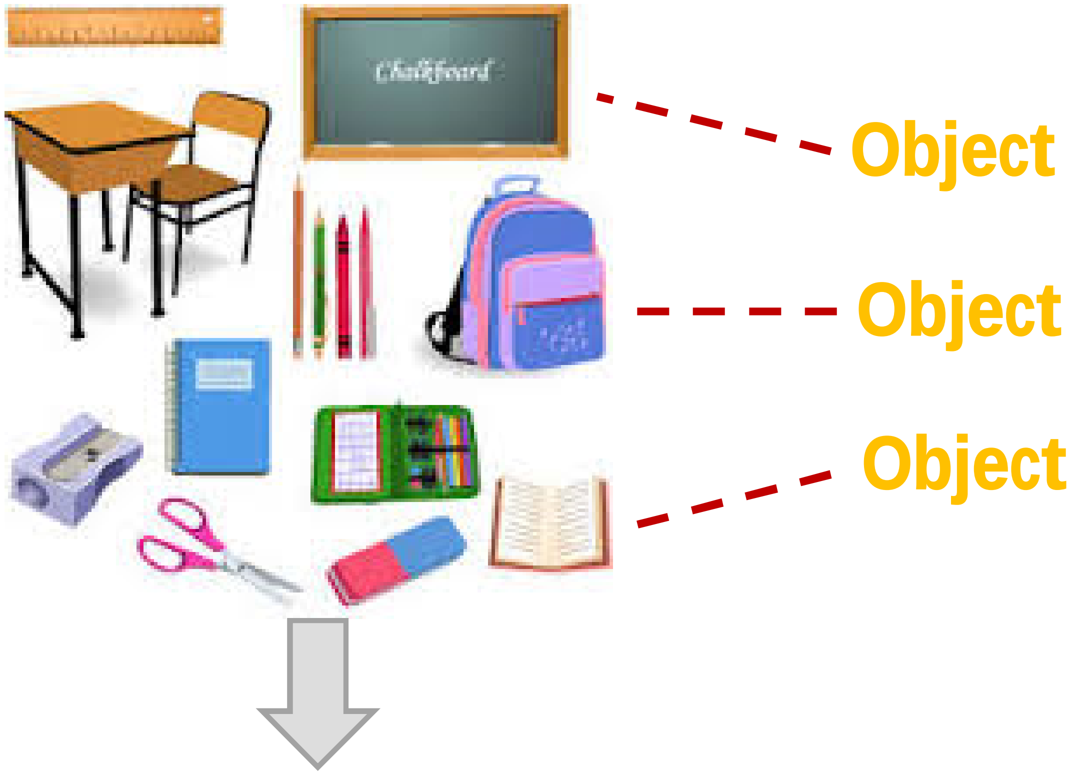
Outline

- Objects and classes revisited.
- Anatomy of Java class.
- Objects and object reference variables.
- Declaring and creating objects in Java
- Messages to an object and methods
- Anatomy of Java method.
- Invoking methods in Java.

Learning Objectives

- To define objects and classes.
- To declare a class.
- To declare data fields/instance variables and methods.
- To create an object.
- To invoke methods of a class

What is Object?



Object

Object

Object

Things that represent entities in the real world

Revision...

What is Object?

has...

Object	Attributes/Properties	Behaviours
 student	name matric No course semester cgpa	set profile update profile print profile register subject print cgpa

Revision...

What is Class?

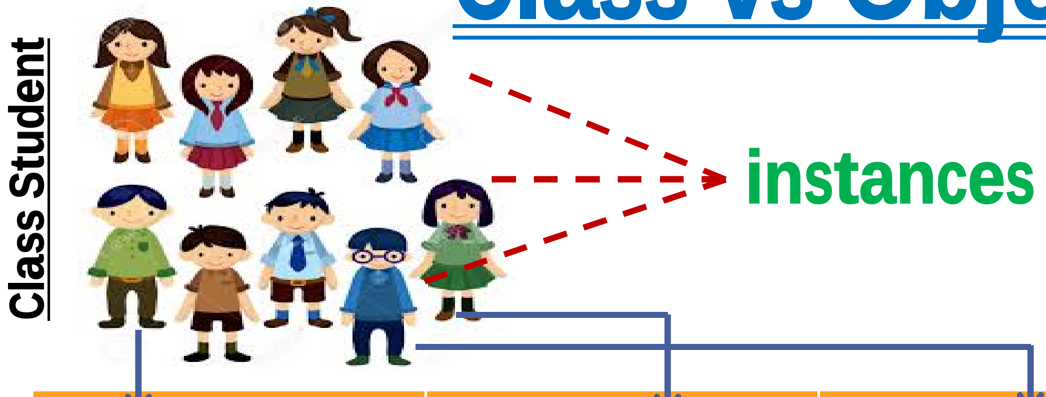
- A class is a category of objects that share the same:
  - Attributes/properties,
  - behaviours

OR

- A class is model, template or blueprint from which objects are created.

Revision...

Class vs Object



student 1	student 2	student 3
name = “Ahmad” matric No = “s1111” course = “IT” semester = 1 cgpa = 3.67 set profile update profile print profile register subject print cgpa	name = “Siti” matric No = “s2222” course = “ACCT” semester = 2 cgpa = 3.01 set profile update profile print profile register subject print cgpa	name = “Zamri” matric No = “s3333” course = “MGNT” semester = 5 cgpa = 2.86 set profile update profile print profile register subject print cgpa

Attributes/Properties

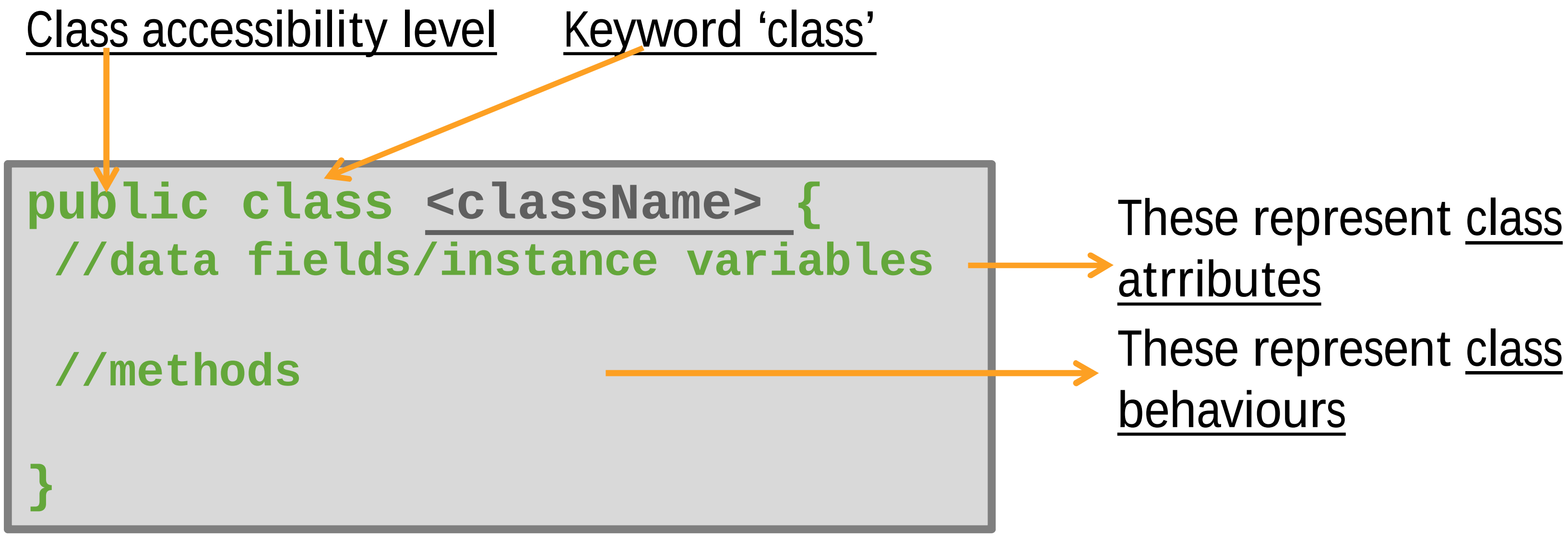
- name
- matric No
- course
- semester
- cgpa

Behaviours

- set profile
- update profile
- print profile
- register subject
- print cgpa

Revision...

Anatomy of Java Class



Class accessibility level

Keyword ‘class’

```
public class <className> {
 //data fields/instance variables

 //methods

}
```

These represent class atrributes

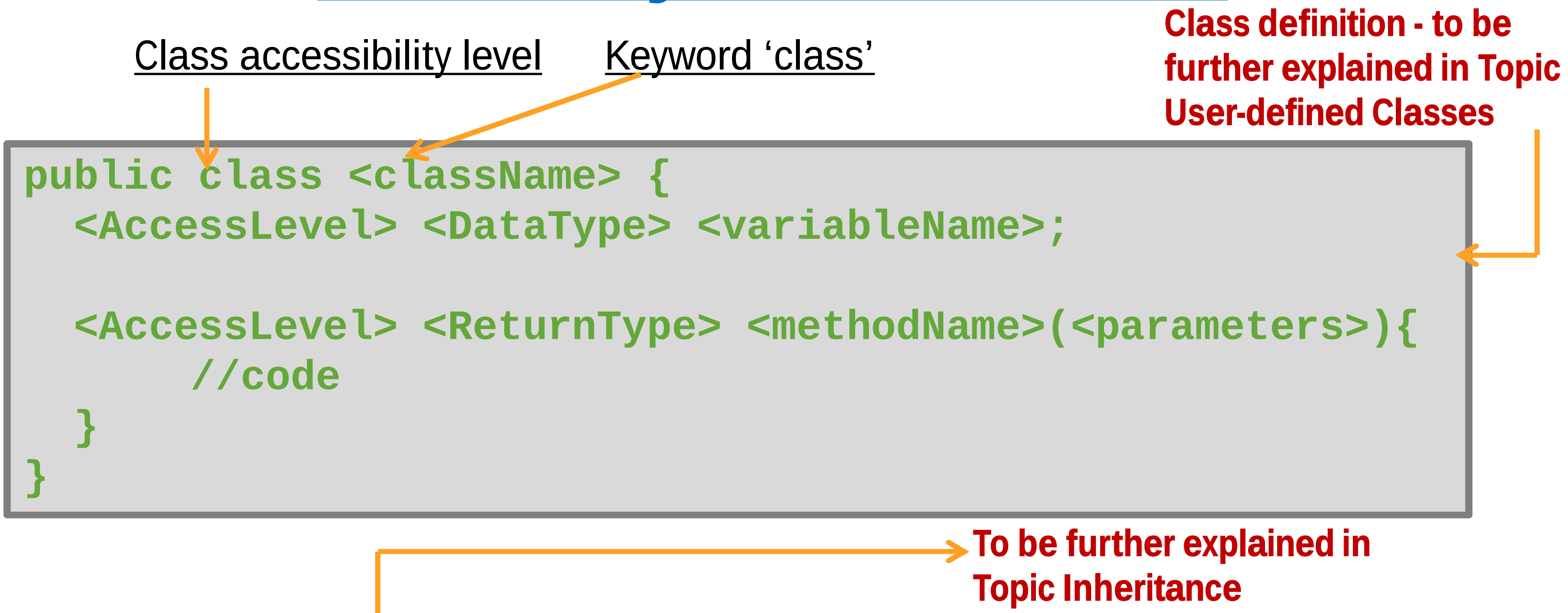
These represent class behaviours

- All the code contained between { } is the **class definition**

Class Name Requirements

- Recommended to begin with an alphabet and a capital letter .
- Can contain only letters, digits, underscore, or dollar sign.
- CANNOT be Java reserved words.

Anatomy of Java Class



Class accessibility level

Keyword ‘class’

Class definition - to be further explained in Topic User-defined Classes

```
public class <className> {
  <AccessLevel> <DataType> <variableName>;

  <AccessLevel> <ReturnType> <methodName>(<parameters>){
       //code
  }
}
```

To be further explained in Topic Inheritance

- Where,
  - AccessLevel – public, private, default, protected
  - DataType – data type of a variable e.g. int, double, String, etc
  - ReturnType – data type returned by the method

Categories of Classes

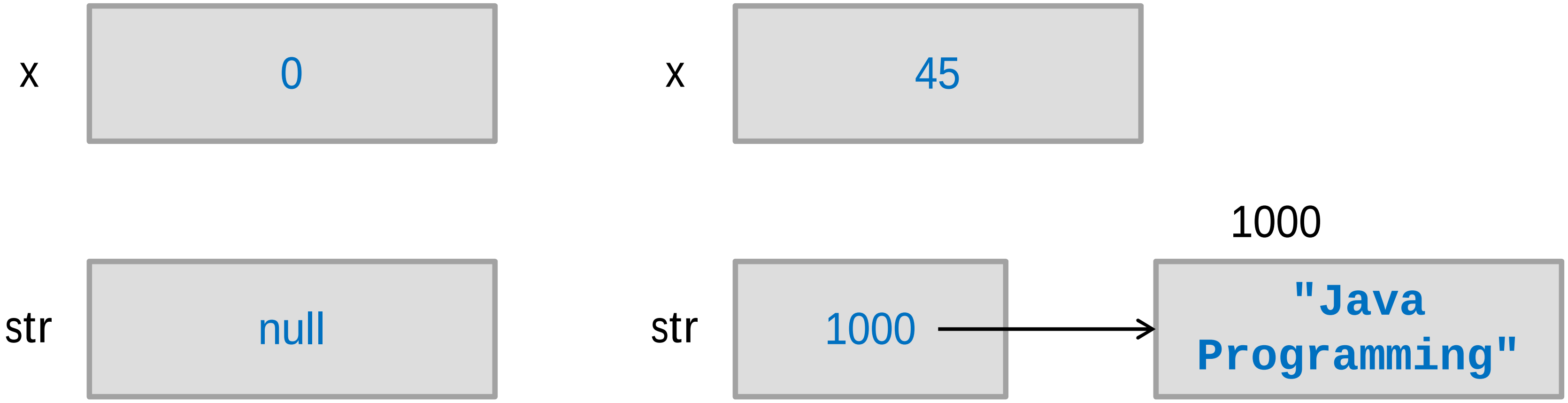
- There are 2 categories of classes:
  - Pre-defined classes → **See Topic Pre-defined Classes**
  - User-defined classes → **See Topic User-defined Classes**

Object & Object Reference Variable

- In Java:
  - Numerical data are called *primitive data types*.
  - Objects are called *reference data types* (or ‘**object reference variable**), because the contents are addresses that refer to memory locations where the objects are actually stored.
- A **variable** can be either a primitive type or an object *reference*.
- A primitive variable contains the value itself.
- An object reference variable holds the address of an object (can be thought of as a pointer to the location of the object).

Primitive vs Object Reference

```
int x;                        // a variable of primitive data type
String str;                   // a variable of an object type. String
                              // is a predefined class
x = 45;                       // store the actual value
str = "Java Programming";  //store the reference to the object
```

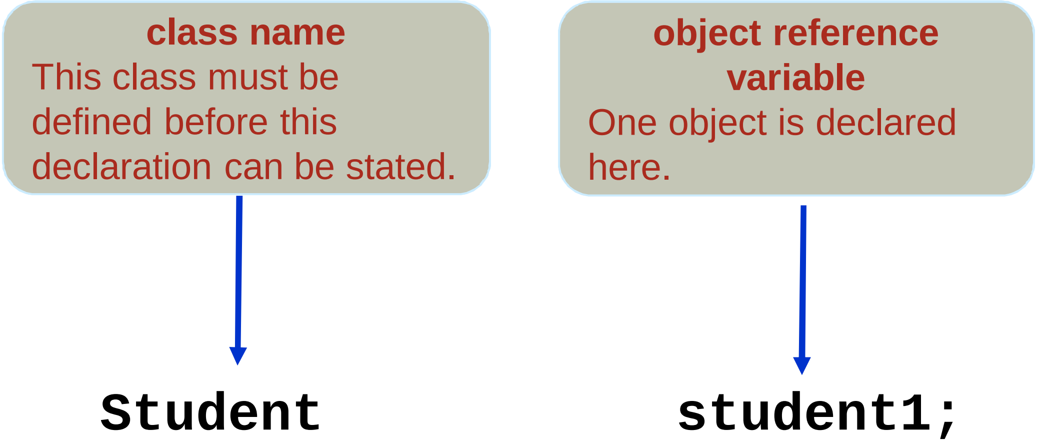


Declaring an Object

- Syntax:

```
<class name> <object reference variable>;
```

- Example:



class name
This class must be defined before this declaration can be stated.

object reference variable
One object is declared here.

```
Student          student1;
```

Note: Also known as declaring of an *object reference*

- More examples:

```
Student student2, student3;
```

Requirements for Naming Object Reference

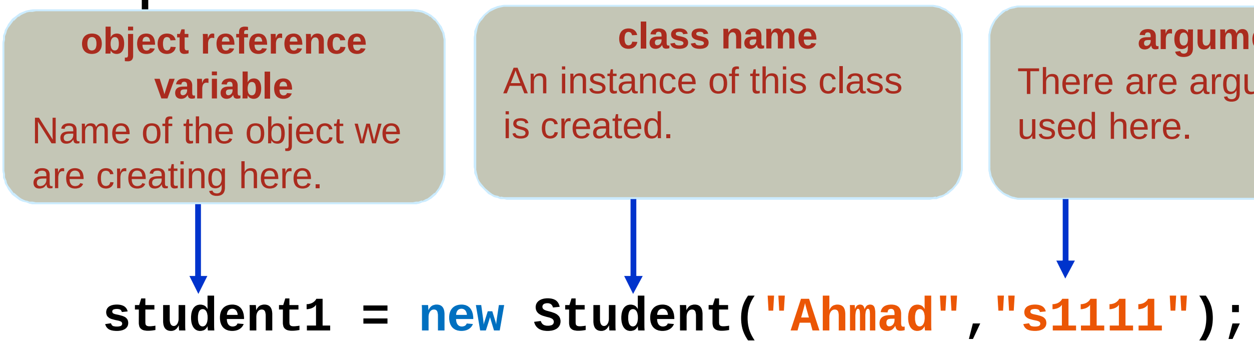
- Can contain only letters, digits, underscore, or dollar sign.
- CANNOT be Java reserved words.
- Recommended to begin with an alphabet and a small letter (if more than 1 words, the first letter of each word is recommended to begin with a capital letter).

Creating an Object

- Syntax:

```
<object reference variable> = new <class
name>(<arguments>);
```

- Example:



object reference variable
Name of the object we are creating here.

class name
An instance of this class is created.

arguments
There are arguments used here.

```
student1 = new Student("Ahmad","s1111");
```

Note: Also known as assigning an *object reference* to the object

- More examples:

```
student2 = new Student("Siti","s2222");
student3 = new Student("Zamri","s3333");
```

Declaring + Creating an Object

- Syntax:

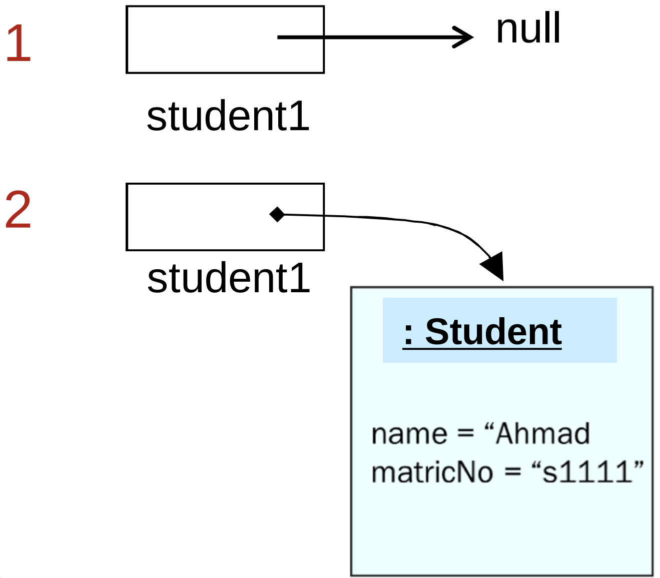
```
<class name> <object reference variable> = new
<className>(<arguments>);
```

- Example:

```
Student student1 = new Student("Ahmad","s1111");
```

Declaring vs Creating an Object

1
```
Student student1;
```
2
```
student1=new Student("Ahmad","s1111");
```



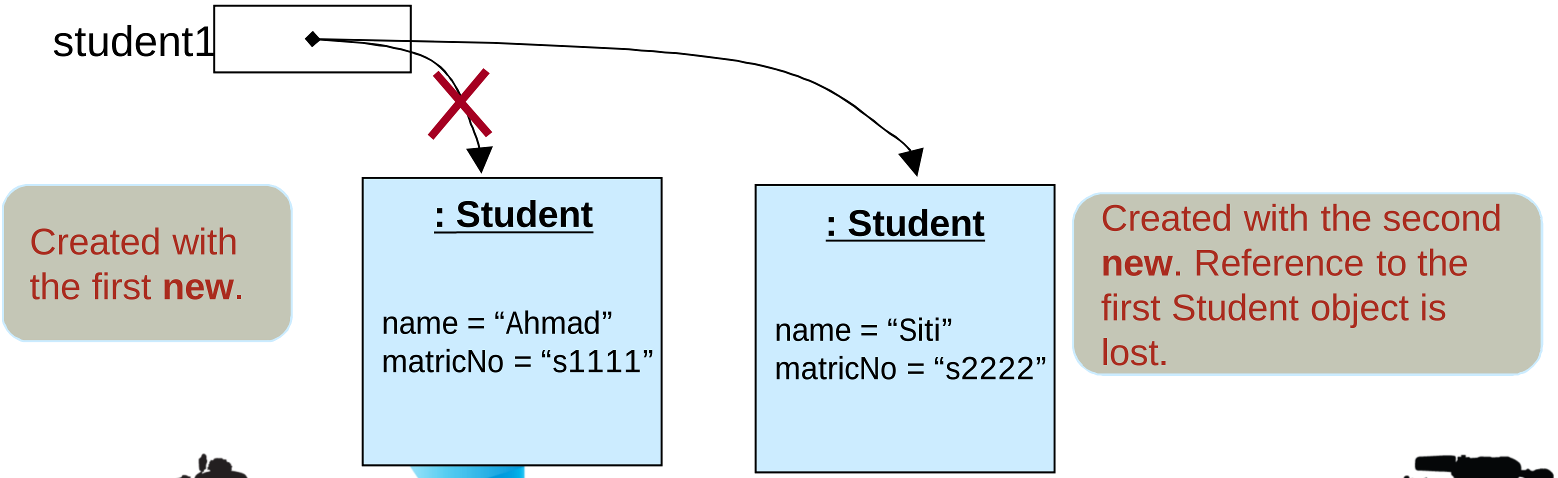
1. An object reference variable named ‘student1’ is declared and space is allocated in memory.

2. A Student object is created and the object reference variable ‘student1’ is set to refer to the object.

Declaring vs Creating an Object

More examples:

```
Student   student1;
student1 = new Student("Ahmad","s1111");
student1 = new Student("Siti","s2222");
```



Created with the first **new**.

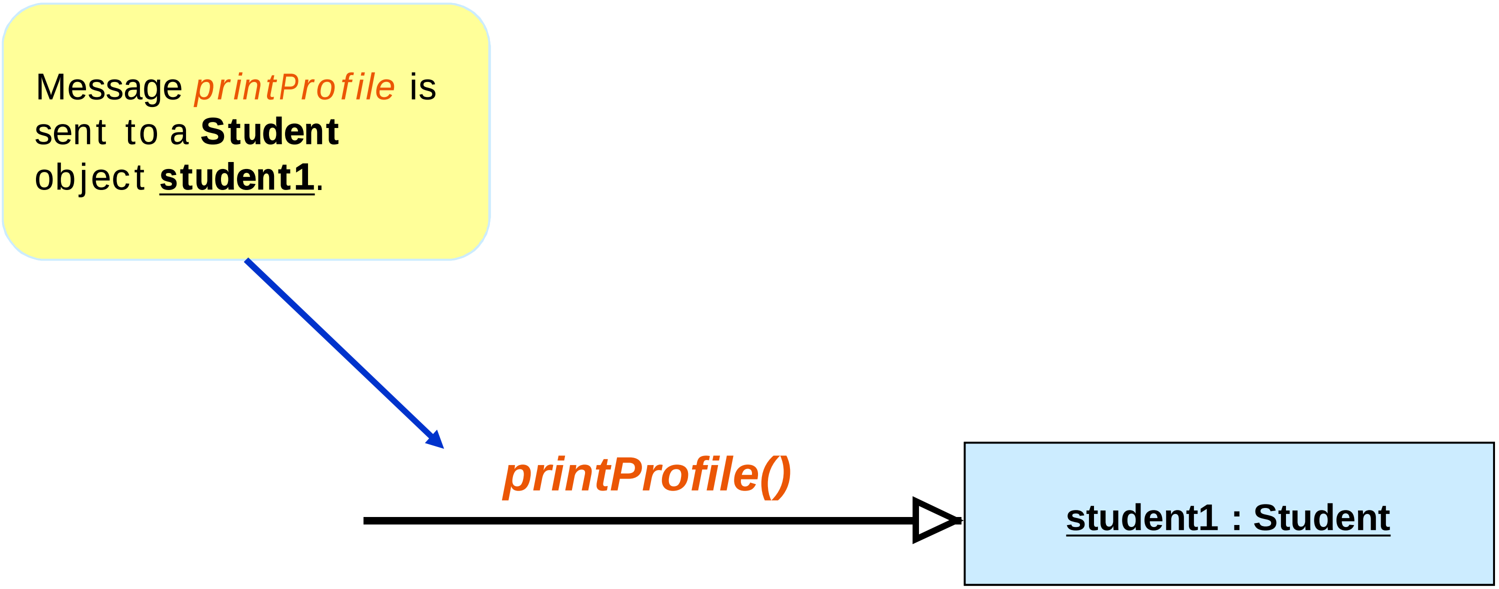
Created with the second **new**. Reference to the first Student object is lost.

Object, Message and Method

- To instruct an object to perform a task, we send a *message* to the object.
- You can **ONLY** send a message that the objects can understand.
- An object must possess a matching *method* to be able to handle the received message. (Note - The *behaviours* of an object implemented in classes are ***methods***)

Object, Message and Method

Message *printProfile* is sent to a **Student** object **student1**.



Object, Message and Method

- After an object is created, its data field can be accessed and its methods can be invoked using the dot operator(.).

```
student1.semester;
```

- access a data field in the object.

```
student1.printProfile();
```

- invokes a method on the object.

Anatomy of Java Method

```
<AccessLevel> <ReturnType> <methodName> (<parameters>){
    //statements
}
```

To be further explained in Topic Inheritance

- Where,
  - AccessLevel – public, private, default, protected
  - ReturnType – data type returned by the method

- All the code contained between { } is the **method definition** (further details will be explained in Topic User-defined Class)

Invoking Methods

- Syntax:

```
<objectReferenceVariable>.<methodName>(<arguments>);
```

object reference variable/object name
Name of the object to which we are sending a message.

method name
The name of the message we are sending.

argument (s)
The argument we are passing with the message.

- Example:

```
student1.printProfile()   ;
```

More Examples

```
student1.editProfile("MM", 1, 3.67);
student1.registerSubject("HCI");
```

Summary

- Each object has attributes/properties and behaviours.
- Objects of the similar kind/type are categorised in a class. E.g. all student's objects can be grouped in a class of type `Student`.
- A class is a template/blueprint to create a collection of similar objects.
- In Java, a class can be a pre-defined or user-defined class.
- Objects interact to each other by sending messages in order to perform certain tasks.