Classes and Objects

A recap & going deeper with objects and classes

Produced by: Dr. Siobhán Drohan

(based on Chapter 1, Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling)

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
http://www.wit.ie/

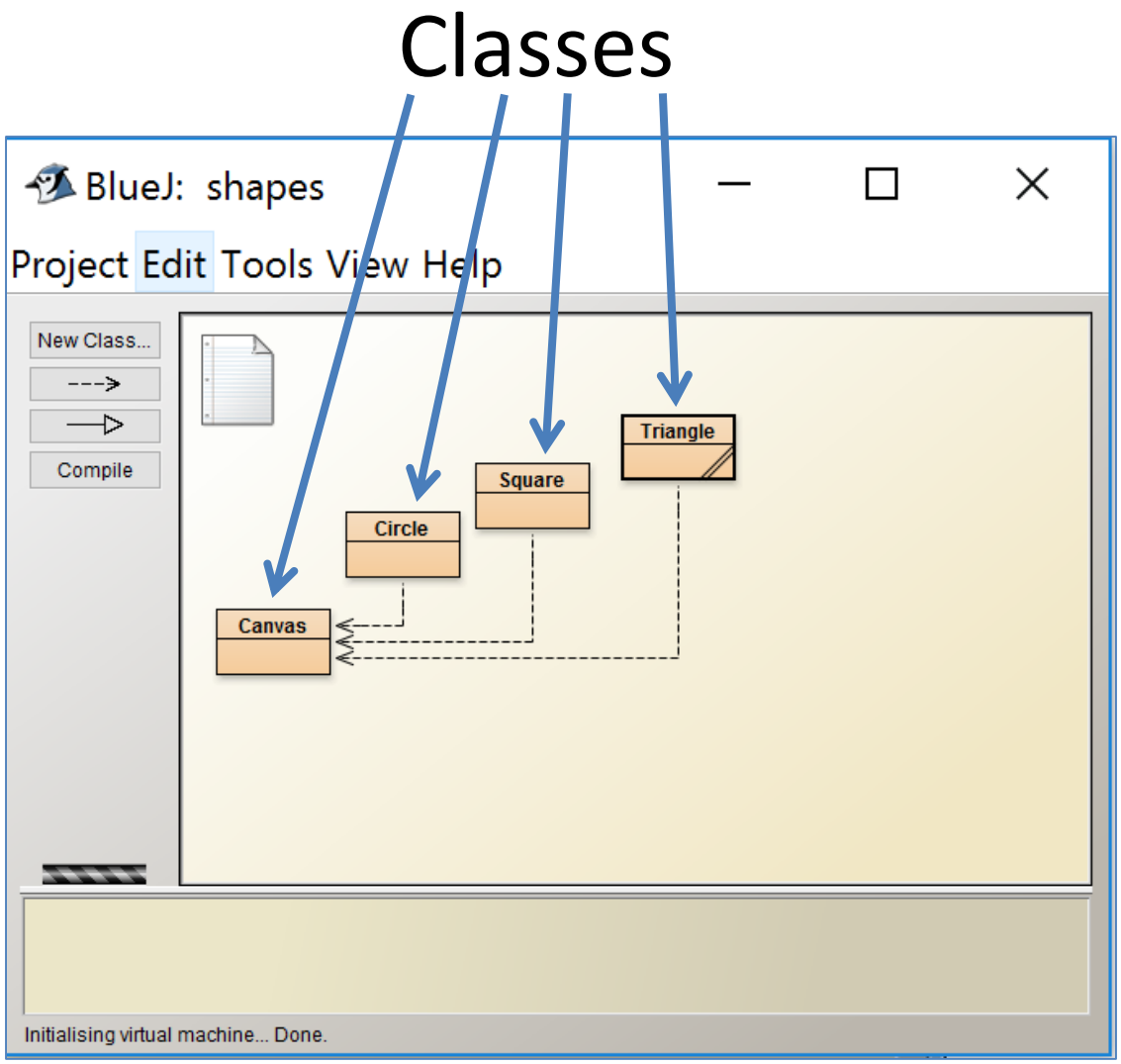
Topic List

- Recap
  - Classes and objects
  - Methods
  - Parameters
  - Data Types
  - Multiple Instances
  - Object State
  - Object Interaction
  - Files in Java
  - JVM
- New Material:
  - Demo: lab-classes
  - Constructors with Parameters
  - Visibility / Access modifiers
  - Objects as parameters
  - Method signature
  - Return Types
  - Naming conventions for Java Classes

Java is an object-oriented language

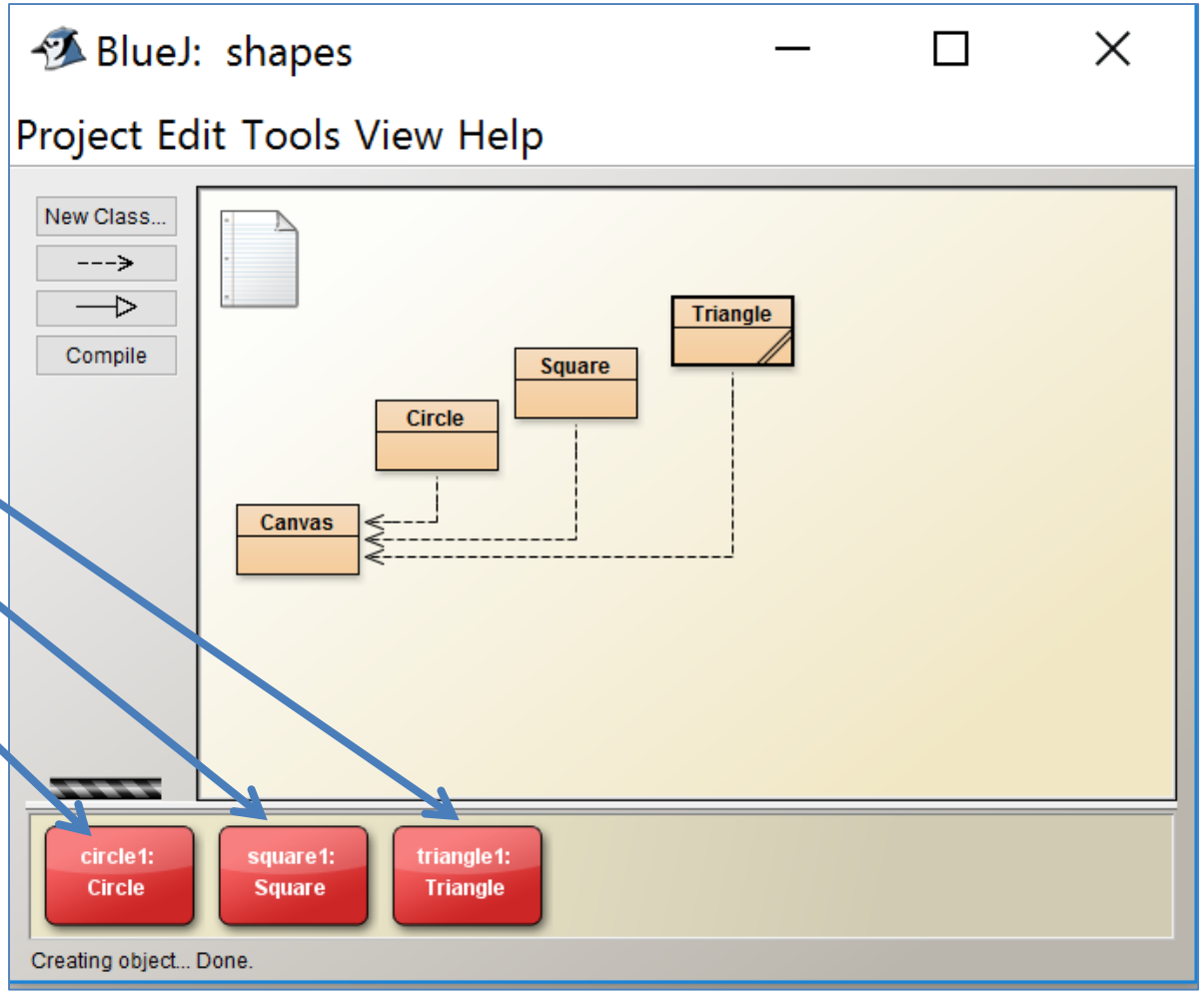
- Modelling some part of the world built up from objects that appear in the problem domain.

- These objects must be represented in the computer model being created e.g.
  - Student
  - Course
  - Teacher



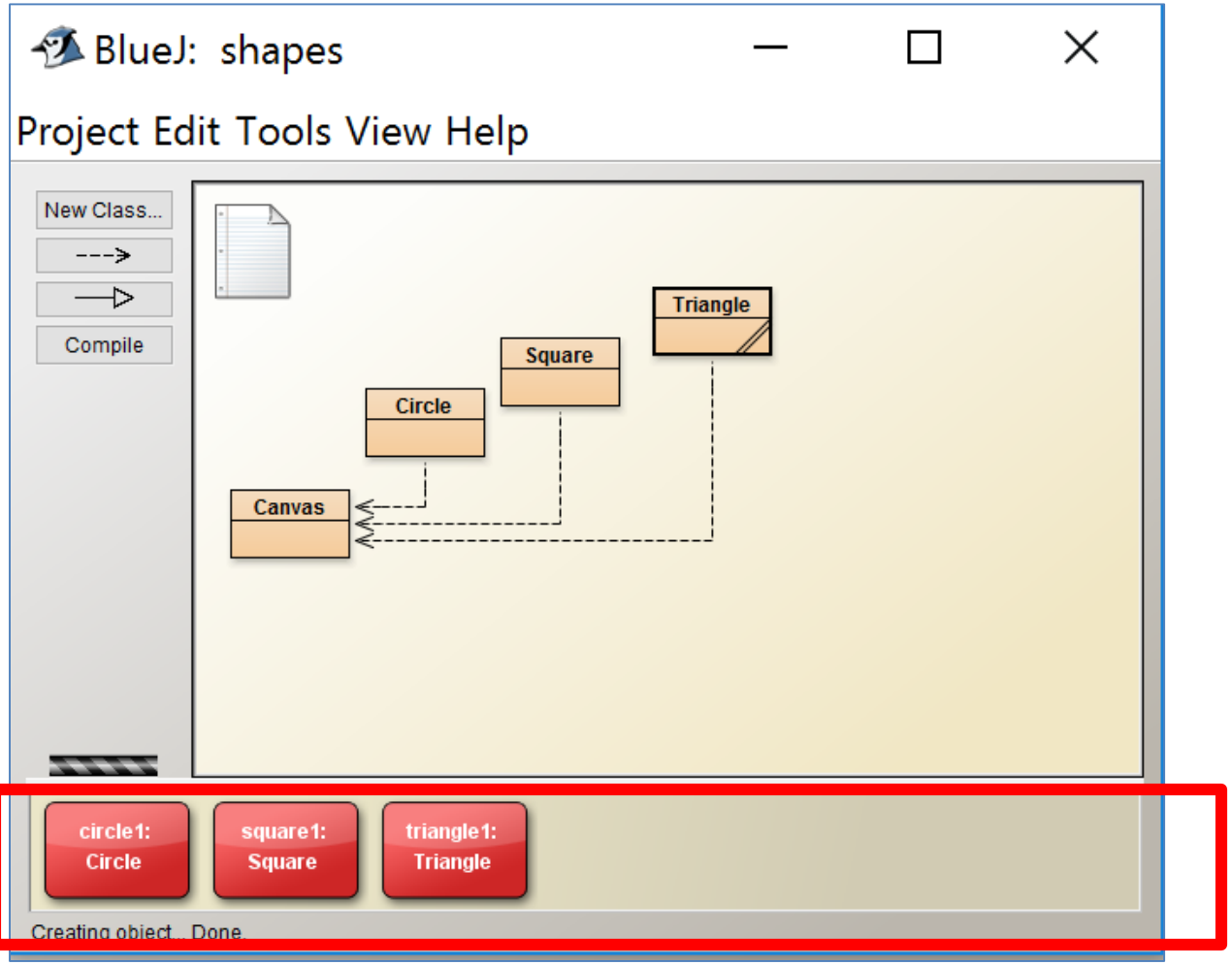
- Represent all objects of a kind e.g.: “car”
- The class describes the kind of object; the class is a template/blueprint.

Objects



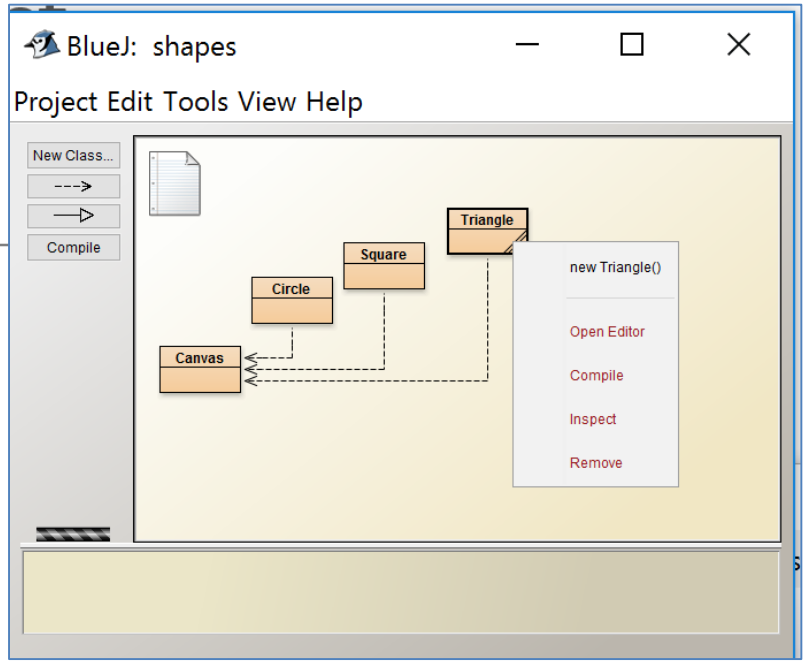
- Objects are created from classes; an object is an instance of a class.
- Represent ‘things’ from the real world, or from some problem domain e.g. “the red car in the car park”.
- The objects represent individual instantiations of the class.

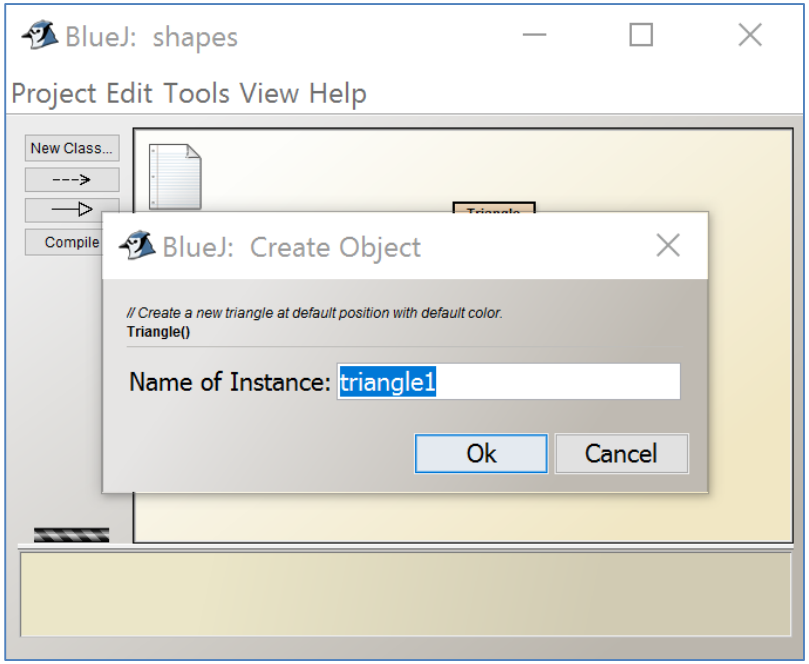
Object bench



Creating an object

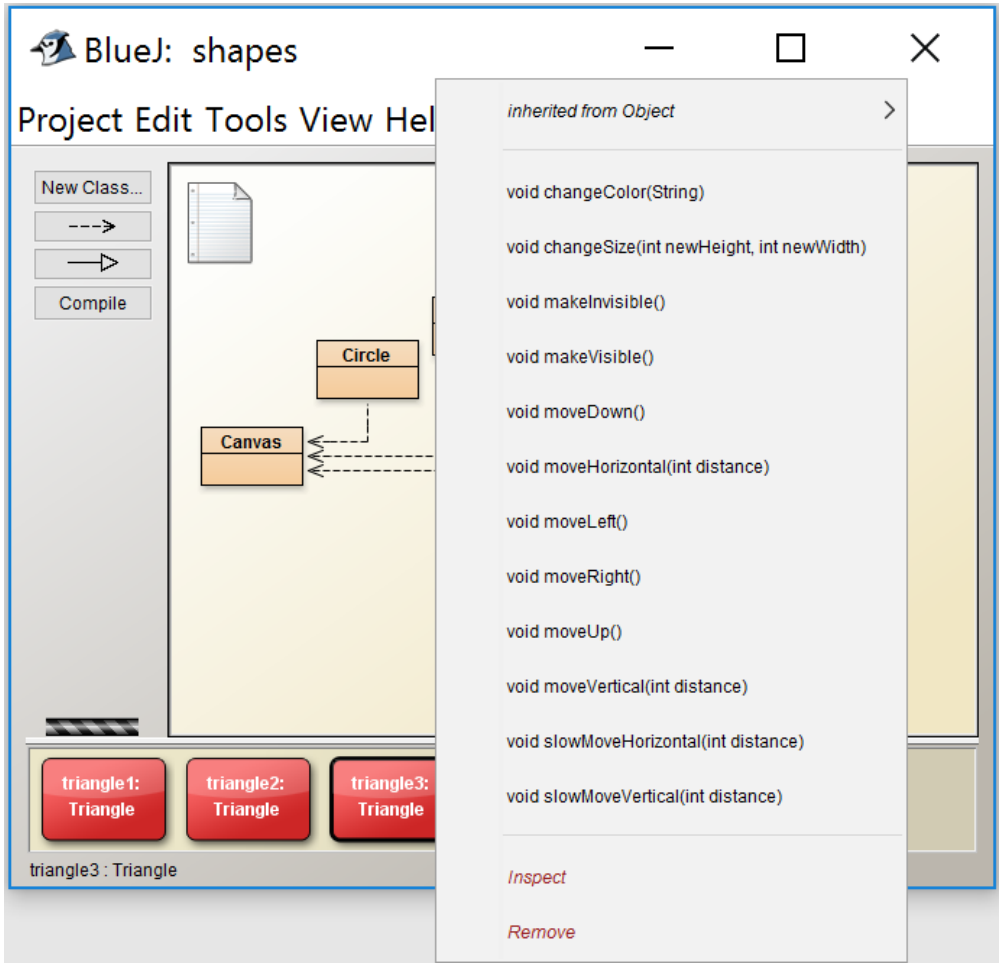
- Right click on the **class**
- From the popup menu, call the constructor e.g. Triangle()
- The constructor is a special method that is the same name as the class.
- You will be asked for the name of the instance e.g. triangle1.
- The constructor “constructs” the object i.e. creates an instance of the class.





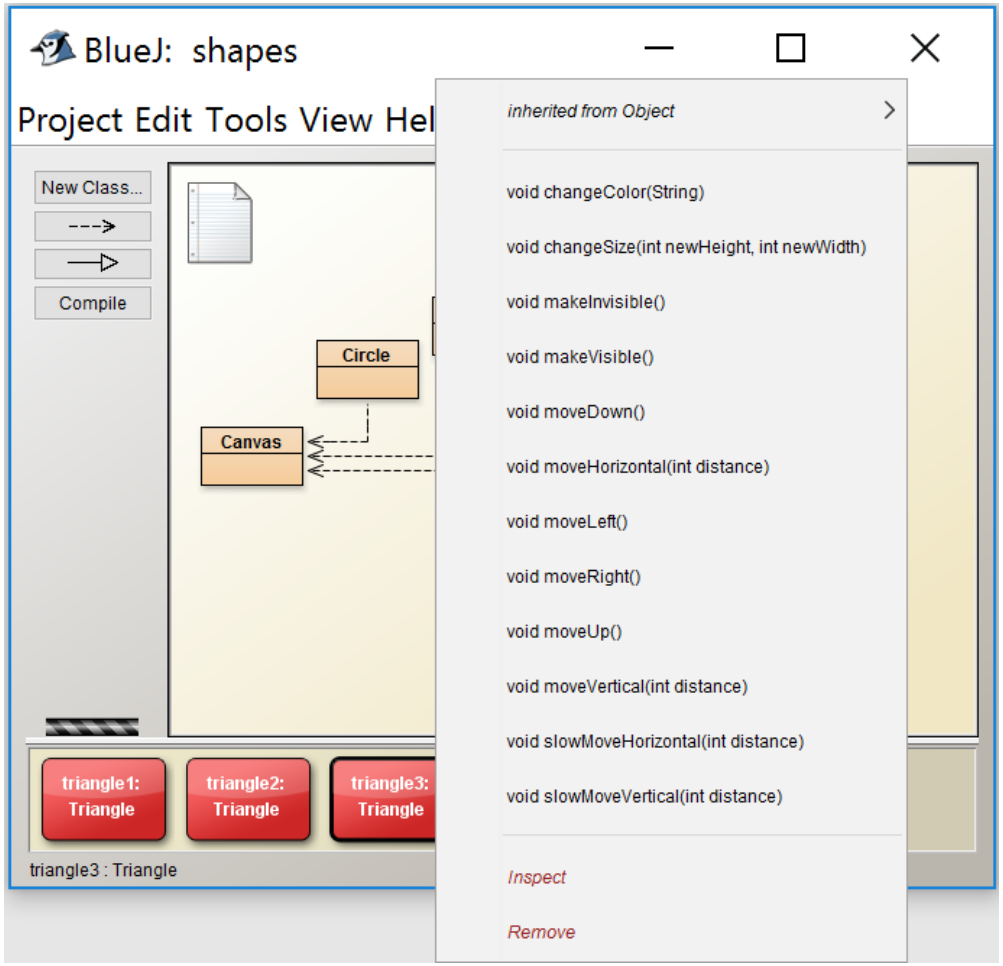
Methods

Objects have operations which can be invoked (Java calls them *methods*).



Calling methods (invoking)

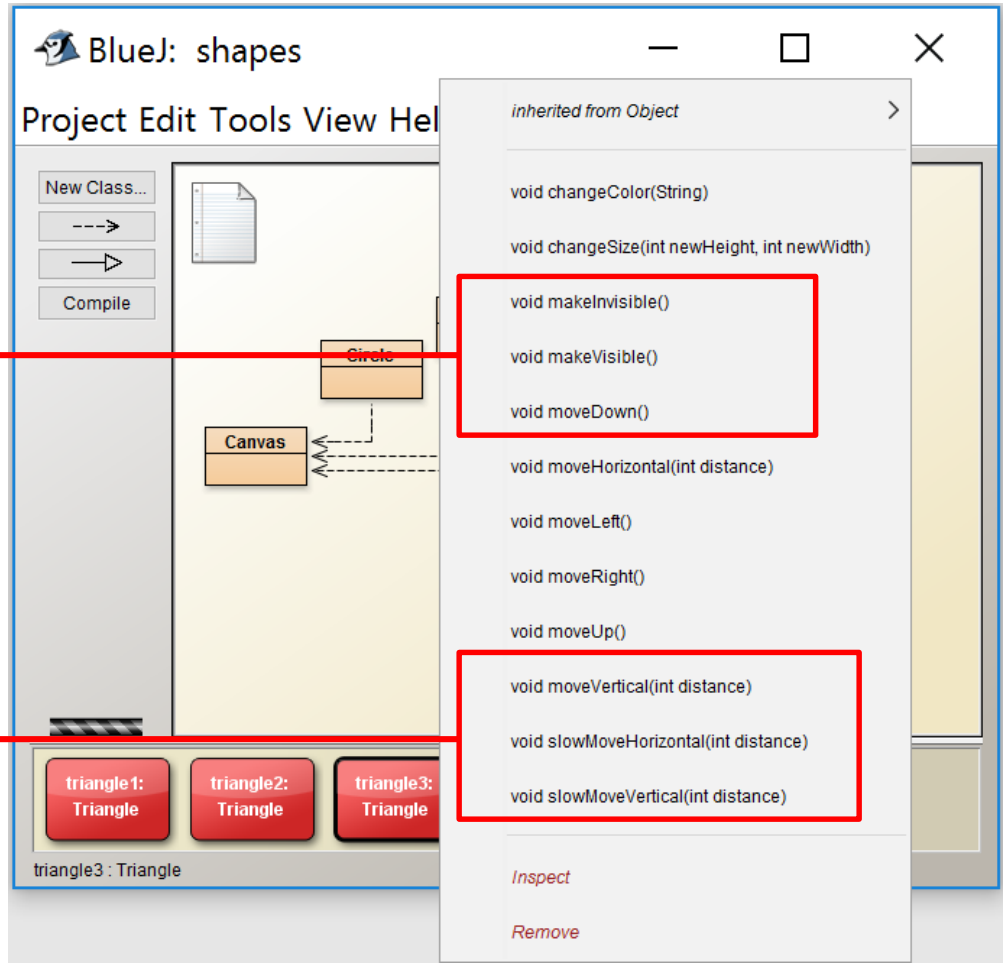
- Right click on the **object**.
- The popup menu lists all the methods that can be invoked on the object.
- Objects usually do something if we invoke a method.
- We can communicate with objects by invoking methods on them.



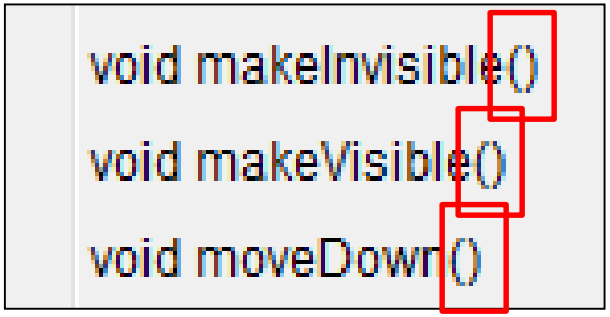
Parameters

These methods have NO parameters

These methods HAVE parameters



Methods with NO parameters



- If the method needs additional information to do its tasks, parameters are typically passed into the method.
- These methods have no parameters as the method doesn't need additional information; note how no variable is passed in the parenthesis i.e. ().

Methods with Parameters

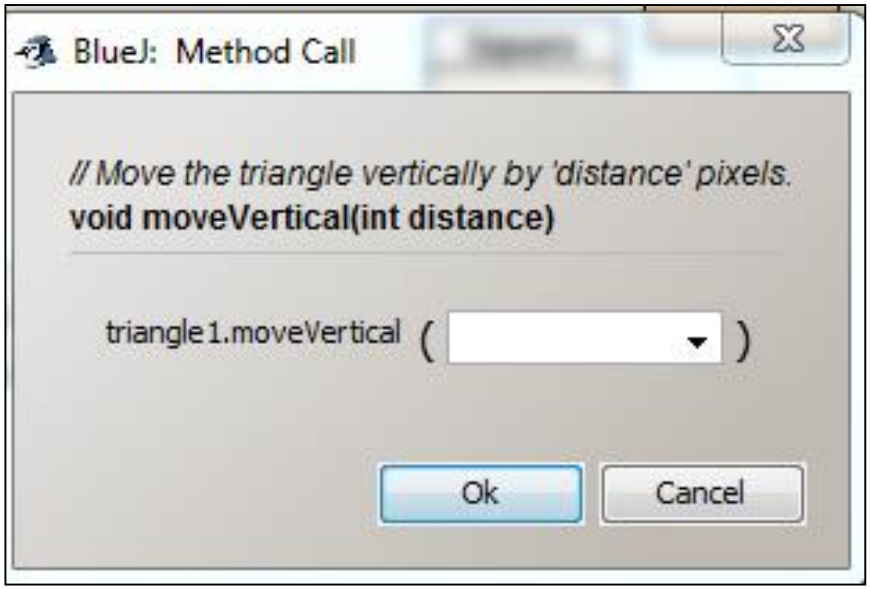
```
void moveVertical(int distance)
void slowMoveHorizontal(int distance)
void slowMoveVertical(int distance)
```

- If a method needs additional information to execute, we provide a parameter so that the information can be passed into it.
- The methods above have one parameter.
- A method can have any number of parameters.
- A parameter is a variable – it has a type (int) and a name (distance).

Methods with Parameters

```
void moveVertical(int distance)
void slowMoveHorizontal(int distance)
void slowMoveVertical(int distance)
```

- In BlueJ, if we invoke the `moveVertical` method, a dialog will pop up asking you to enter a value for `distance`.
- As the `distance` variable is declared as an `int`, we enter a whole number.



Variables

- Variables are used to store information.

- In Java, each variable must be given:
  - A variable name e.g. **distance**
  - A data type e.g. **int**

- We will cover variable name conventions later.

Data types

- When we define a variable, we have to give it a type.
- So far, we have seen three different data types for our variables:
  - `int`
  - `boolean`
  - `String`
- The type defines the kinds of values (data) that can be stored in the variable.

Data types

- **`int`**

  This type holds whole numbers

- **`boolean`**

  This type holds EITHER true or false.

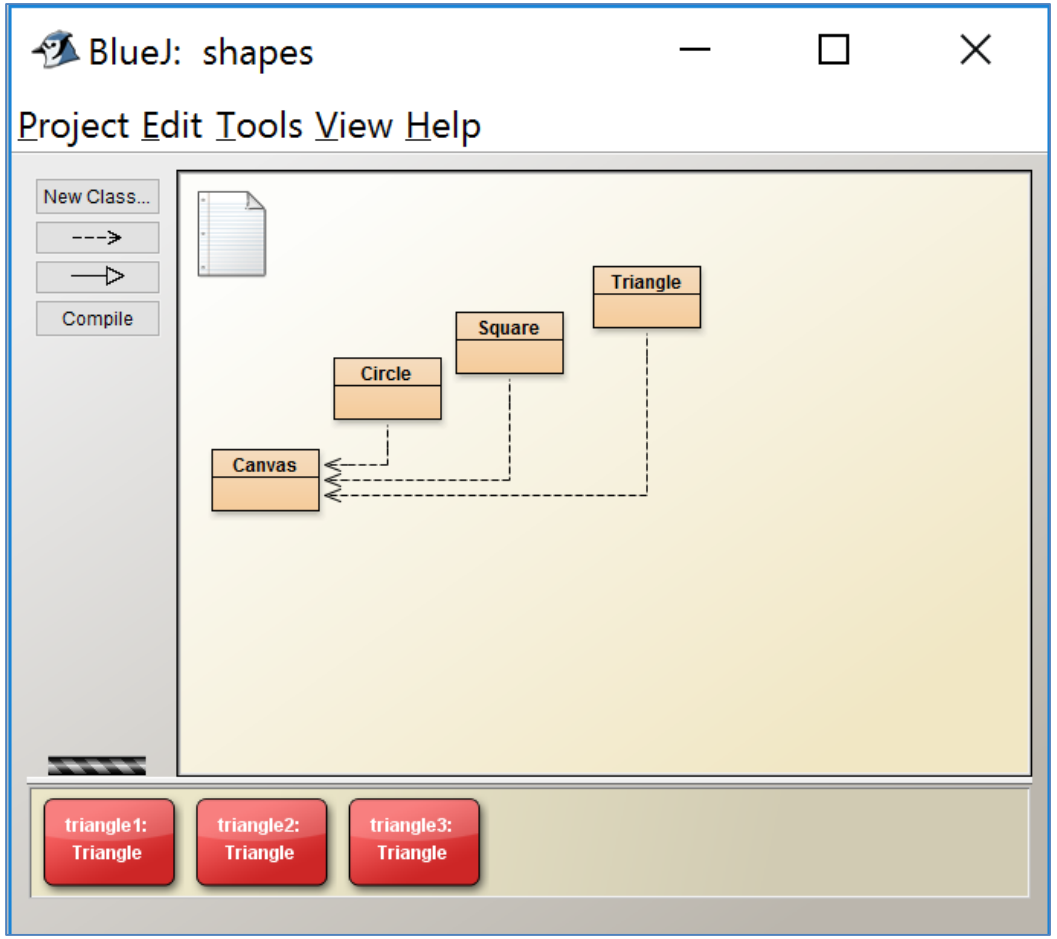
- **`String`**

  This type holds a number of characters.
  Strings are enclosed within “ ”.

There are more data types in Java and we will cover these in due course.

Multiple Instances

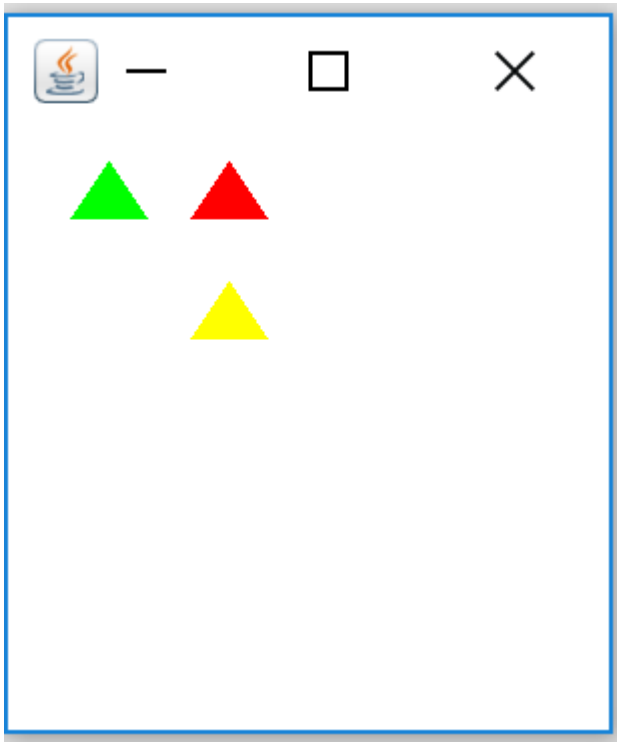
- You can create as many instances (objects) of a class as required.
- In this screen shot, there are three objects (instances) of the Triangle class.



Object State

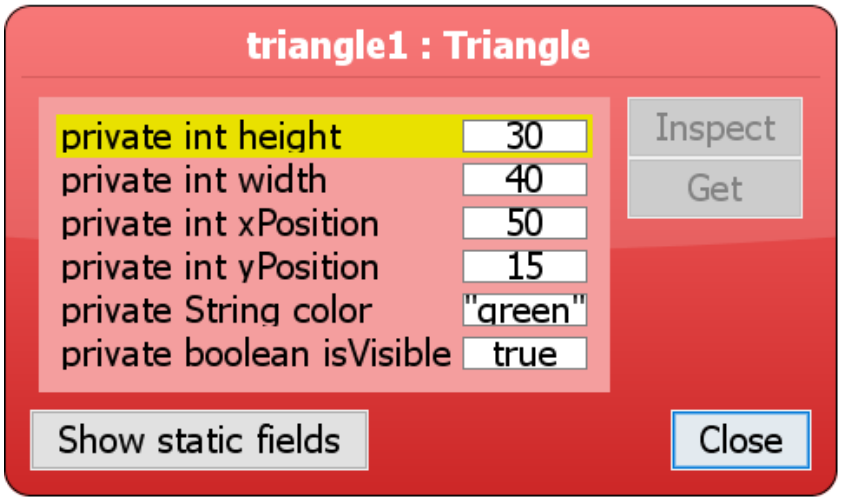
- Each of the Triangle objects on the previous slide has its own state.

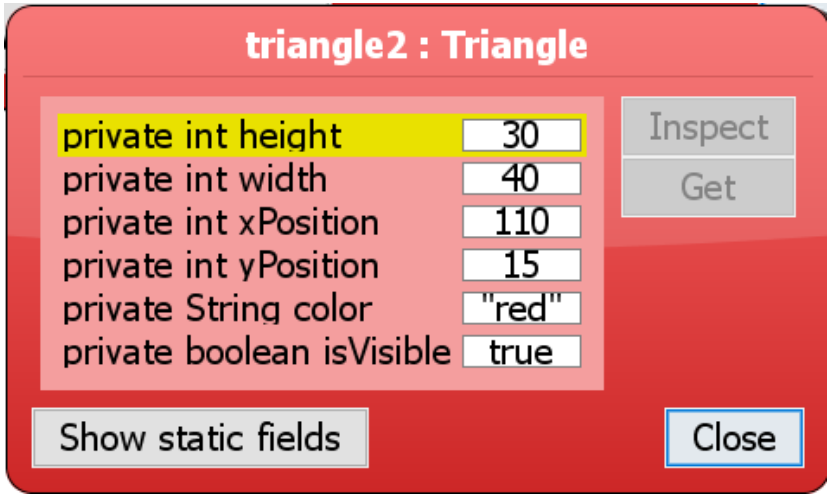
- We can see they are all different colours and have a different position on the canvas.

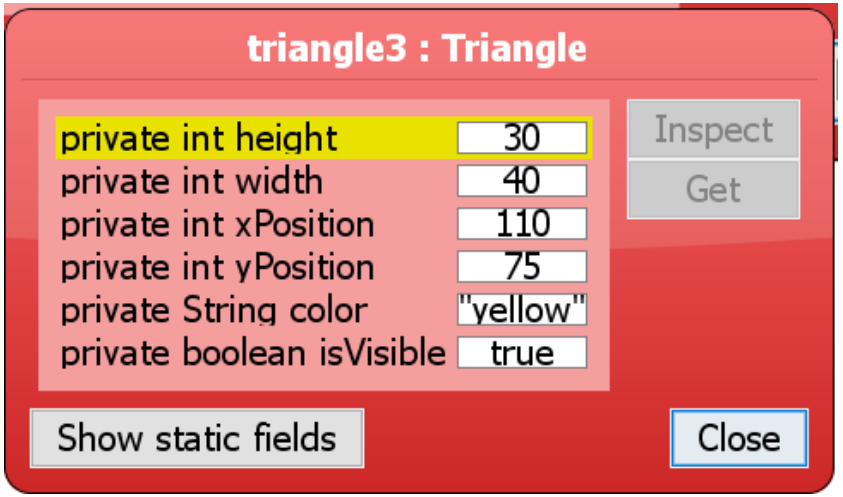


Object State

- In BlueJ, double clicking on the object displays the object state.

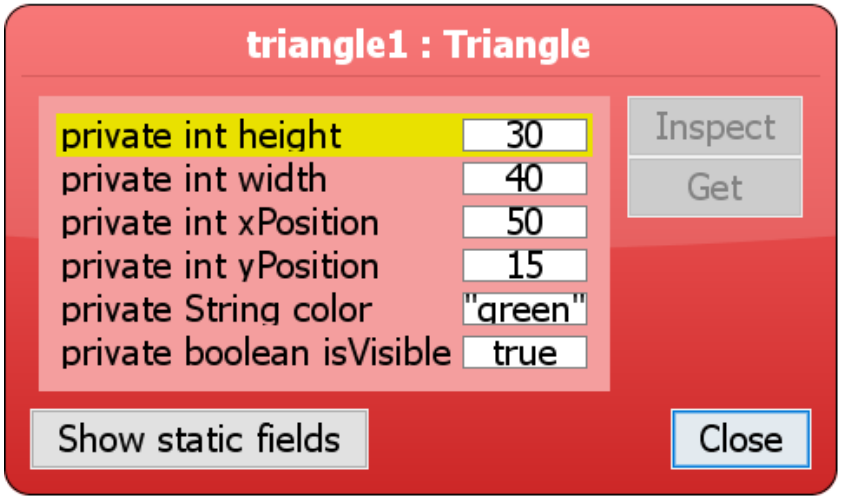


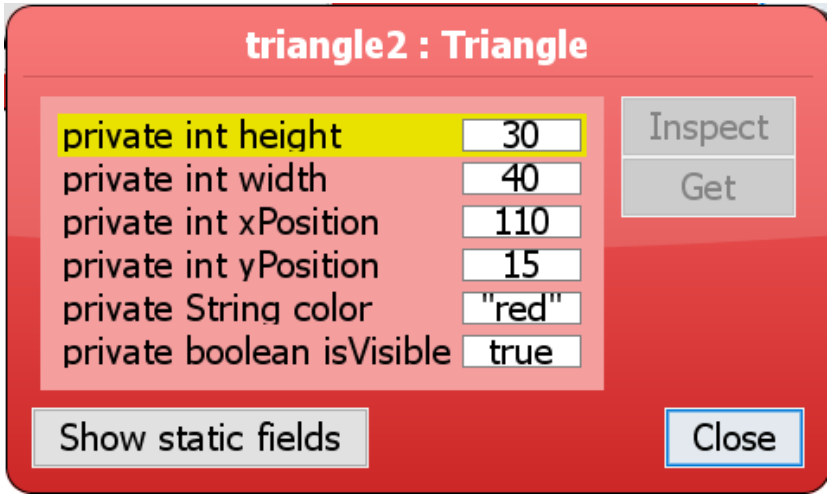


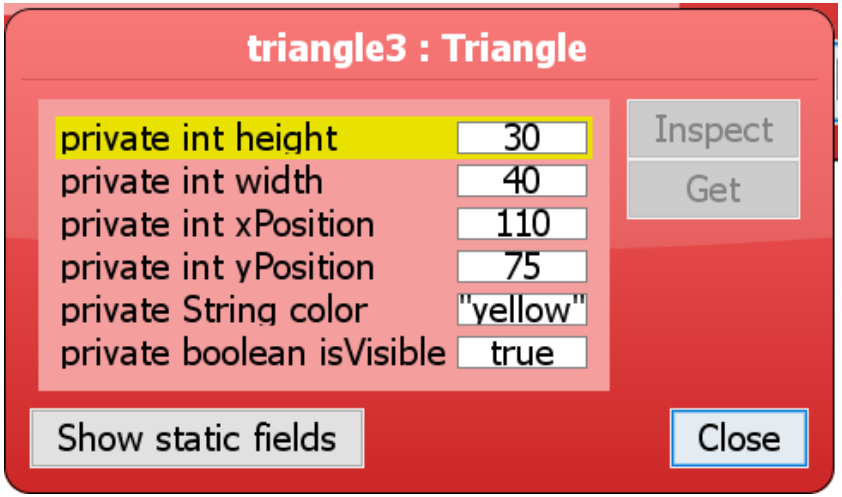


Object State

- An object has *attributes*: values stored in *fields*.
- The class defines what fields (variables) an object has, but each object stores its own set of values (the *state* of the object).

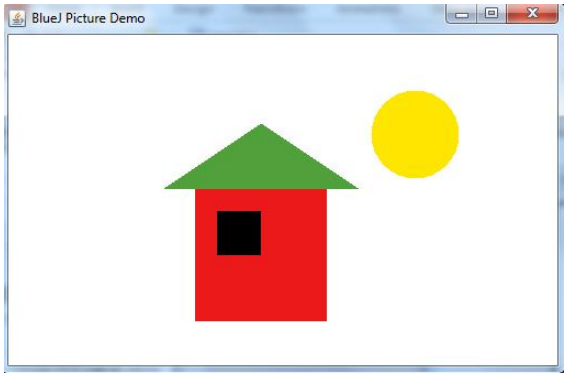






Object Interaction

- In the Picture class, the draw() method creates:
  - Two Square objects
  - One Triangle object
  - One Circle object



- Methods are invoked over these objects to alter their position, change their colour and their size.
- Objects communicate by calling each other's methods.

Topic List

- Recap
  - Classes and objects
  - Methods
  - Parameters
  - Data Types
  - Multiple Instances
  - Object State
  - Object Interaction
  - Files in Java
  - JVM

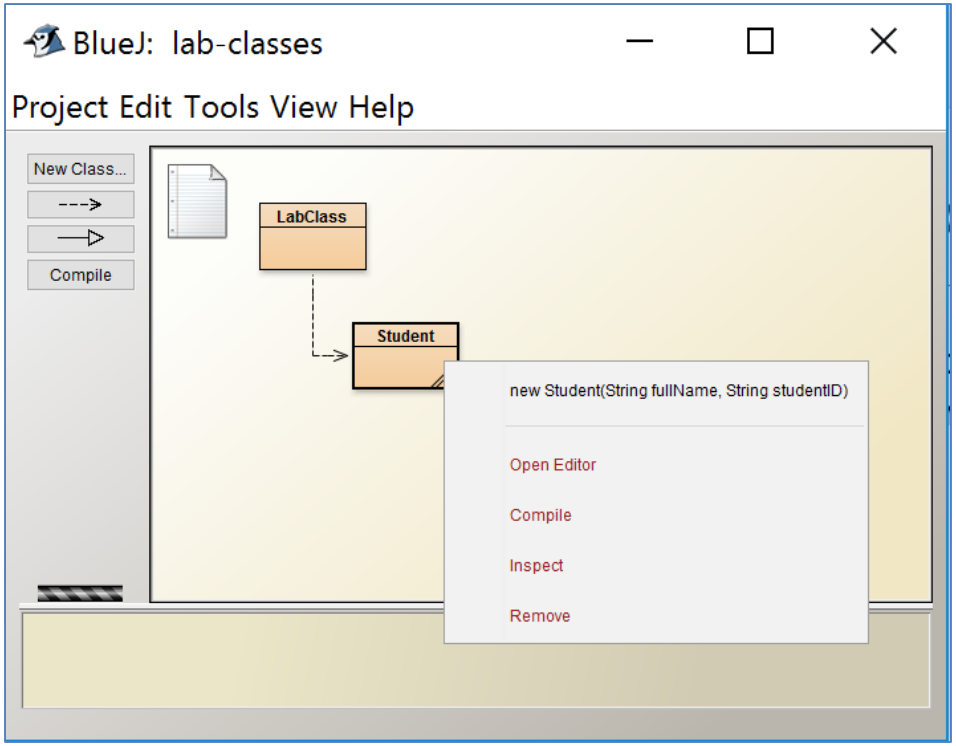
- New Material:
  - Demo: lab-classes
  - Constructors with Parameters
  - Visibility / Access modifiers
  - Objects as parameters
  - Method signature
  - Return Types
  - Naming conventions for Java Classes

Demo

lab-classes project

(source code and file structure)

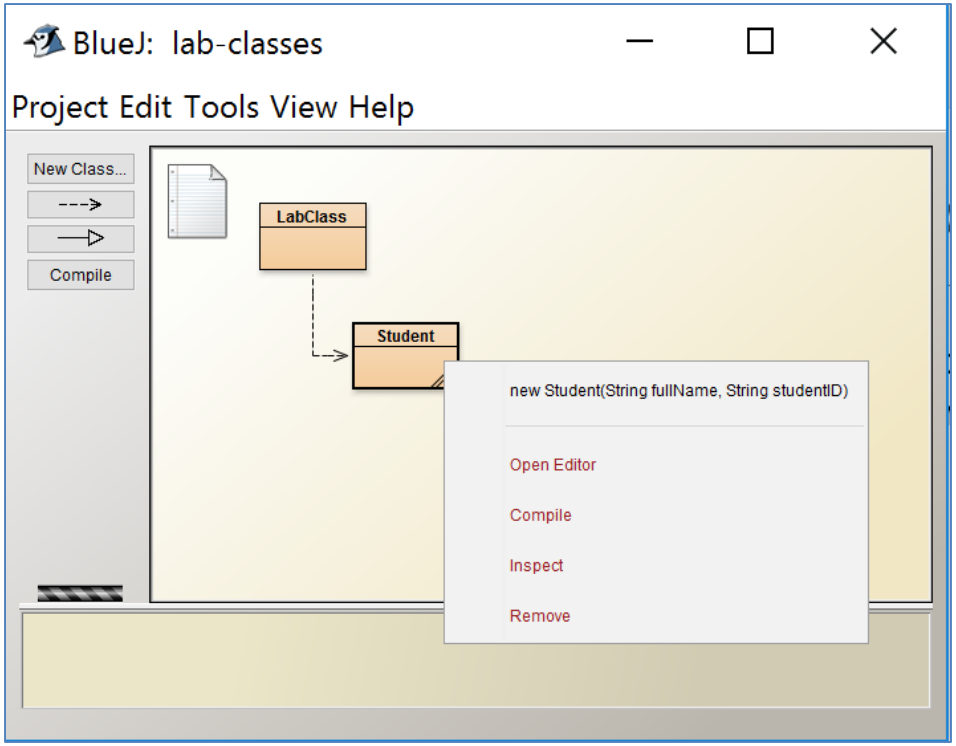
Constructors with parameters



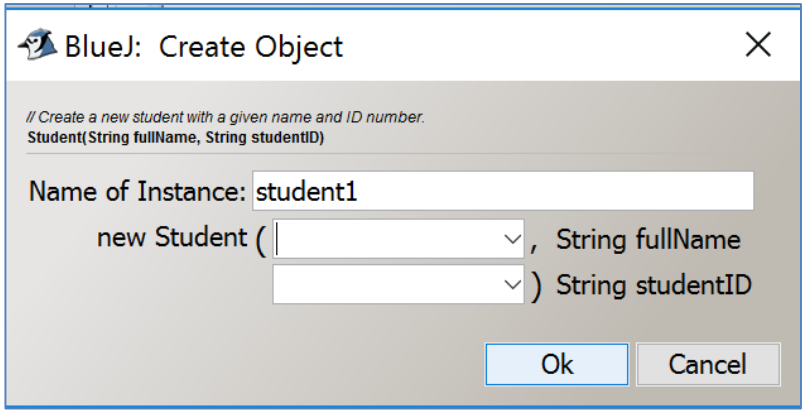
Recap:

- A constructor is a special method that is the same name as the class.
- It “constructs” the object i.e. creates an instance of the class.

Constructors with parameters

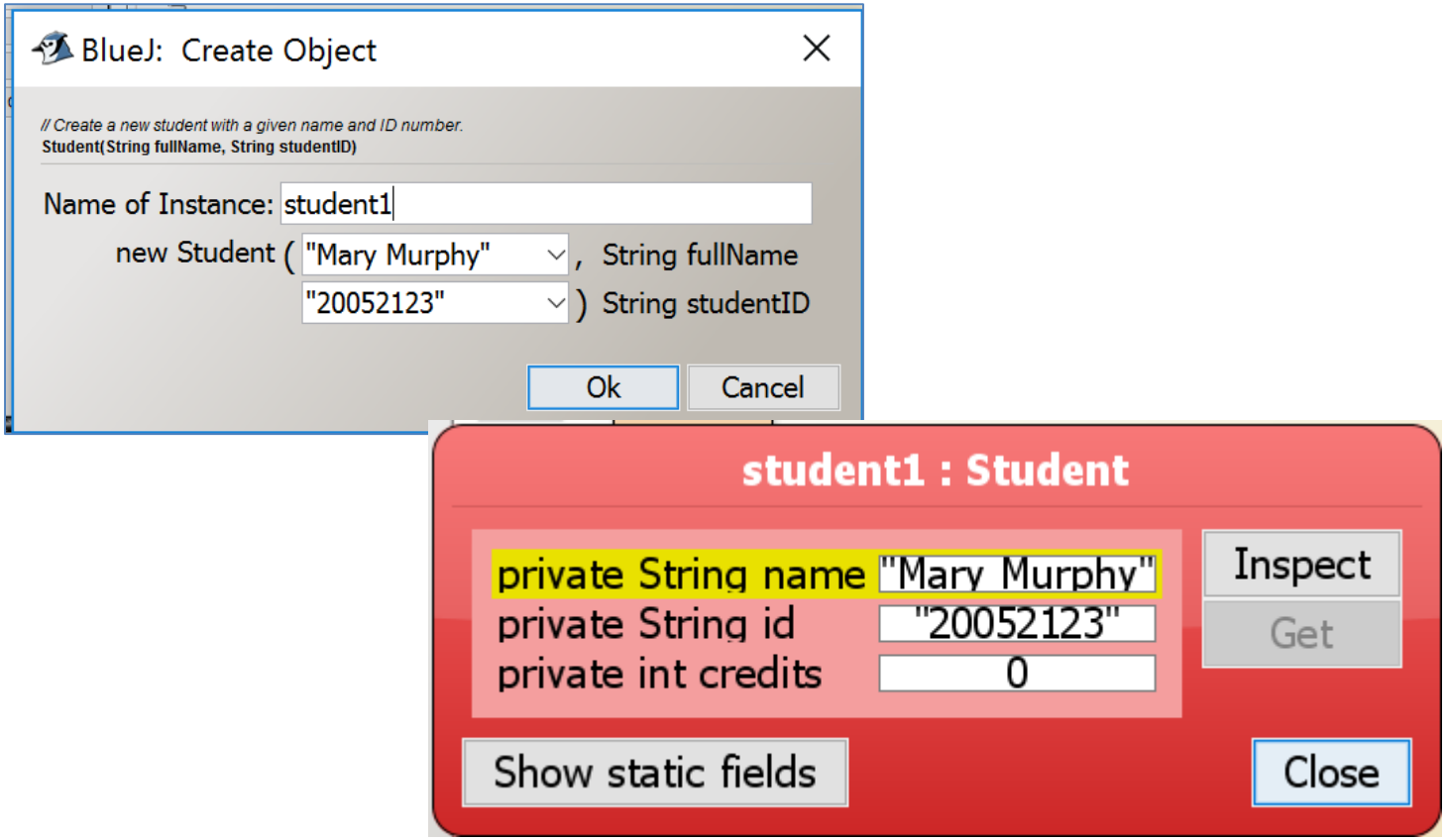


When a constructor with parameters is called, a window will pop up asking you to enter the required information:



Constructors with parameters

The entered information is then used to set up the starting state of the object:



Constructors with parameters

A constructor typically sets a starting state for an object.

Student.java

```
public class Student
{
    // the student's full name
    private String name;
    // the student ID
    private String id;
    // the amount of credits for study taken so far
    private int credits;

    /**
     * Create a new student with a given name and ID number.
     */
    public Student(String fullName, String studentID)
    {
        name = fullName;
        id = studentID;
        credits = 0;
    }
```

Visibility / Access modifiers

```
public class Student
{
    // the student's full name
    private String name;
    // the student ID
    private String id;
    // the amount of credits for study taken so far
    private int credits;
```

Access level modifiers determine whether other classes can use a particular field or invoke a particular method.

https://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html

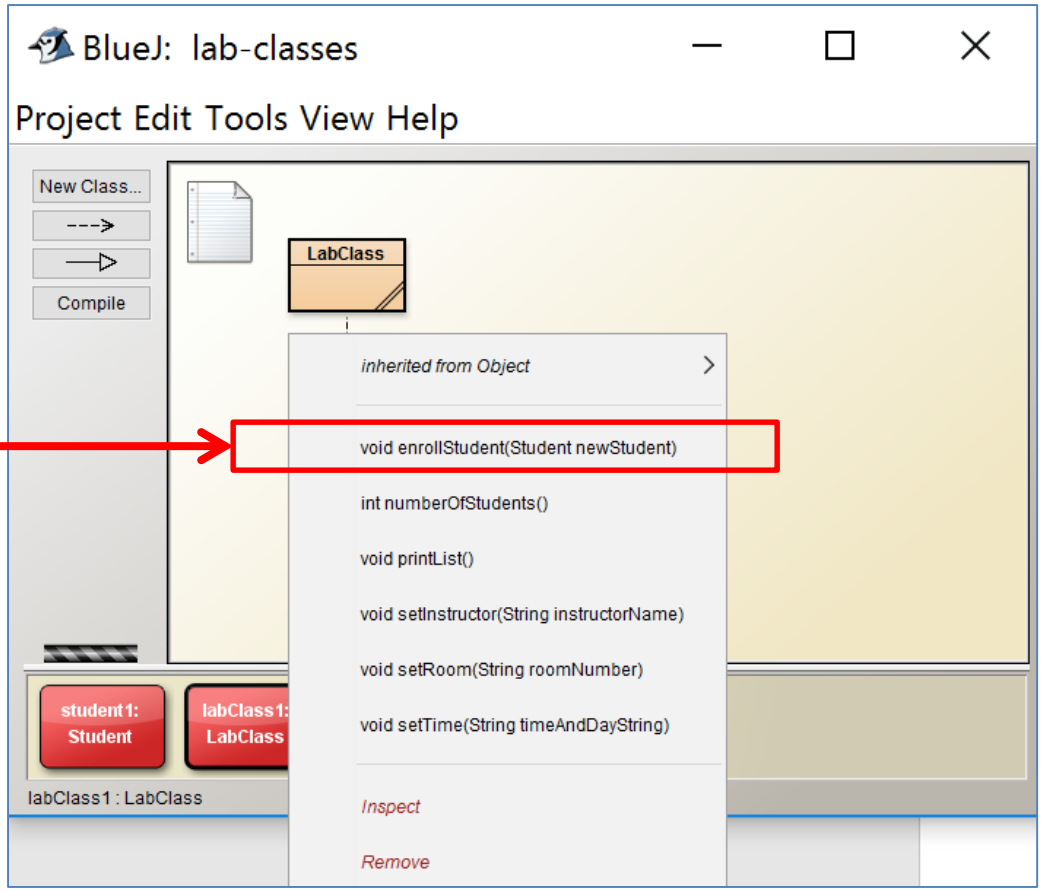
Visibility / Access modifiers

Access Modifiers	Default	private	protected	public
Accessible inside the class	yes	yes	yes	yes
Accessible within the subclass inside the same package	yes	no	yes	yes
Accessible outside the package	no	no	no	yes
Accessible within the subclass outside the package	no	no	yes	yes

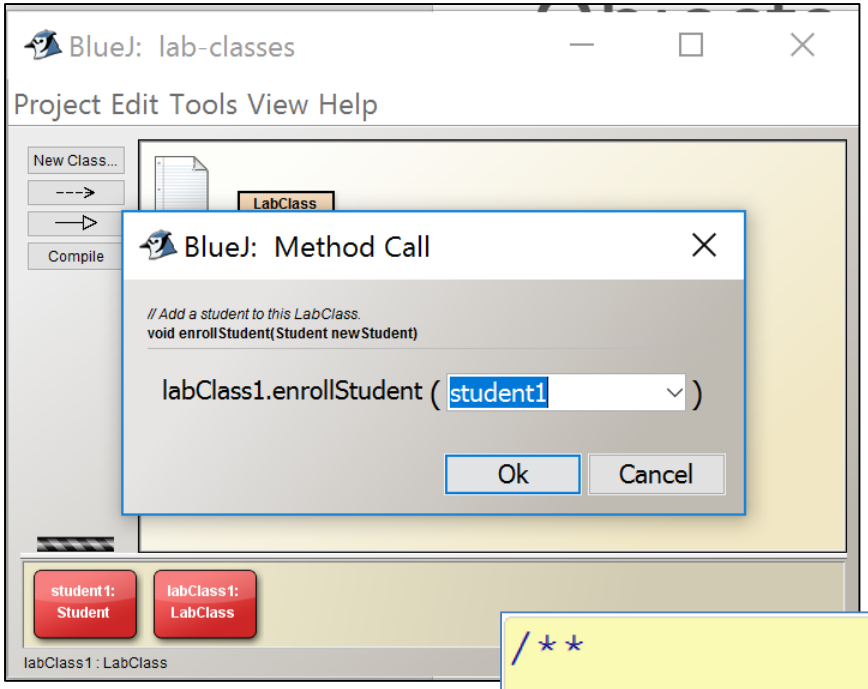
source

Objects as parameters

- Objects can be passed as parameters to methods of other objects.



Objects as parameters



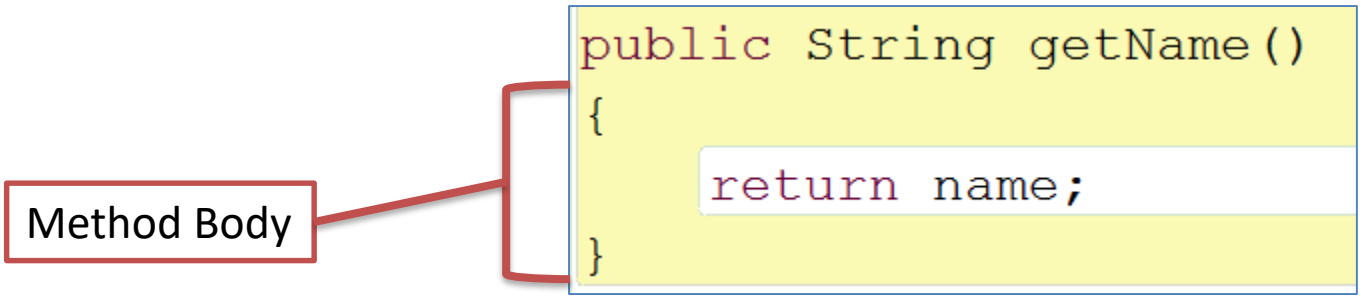

```
/**
 * Add a student to this LabClass.
 */
public void enrollStudent(Student newStudent)
{
    //code ommitted
}
```

Method signature

The method signature consists of a method name and its parameter type list e.g.

getName()
changeName (String)

The method body encloses the method's statements i.e. the code block for the method

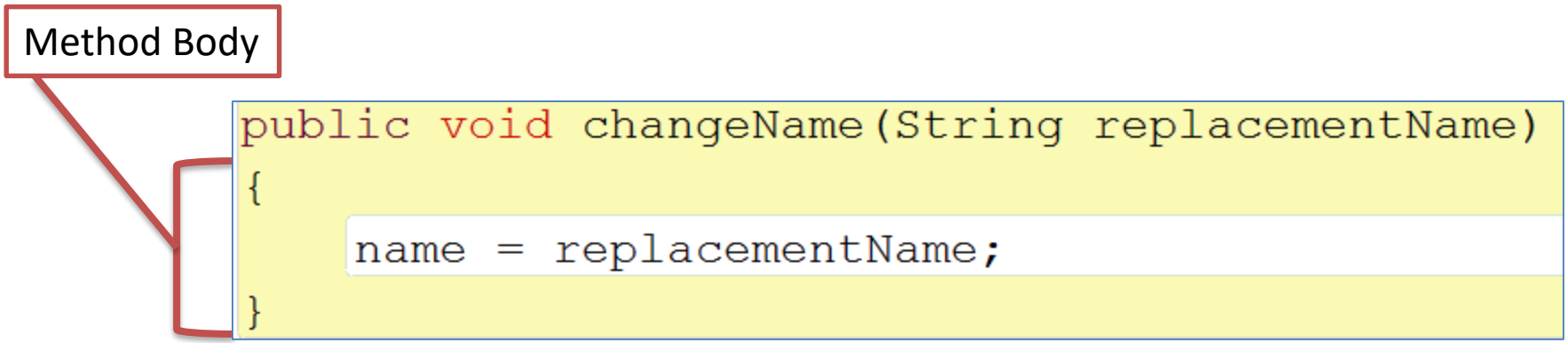


Method signature

The method signature consists of a method name and its parameter type list e.g.

getName()
changeName (String)

The method body encloses the method's statements.

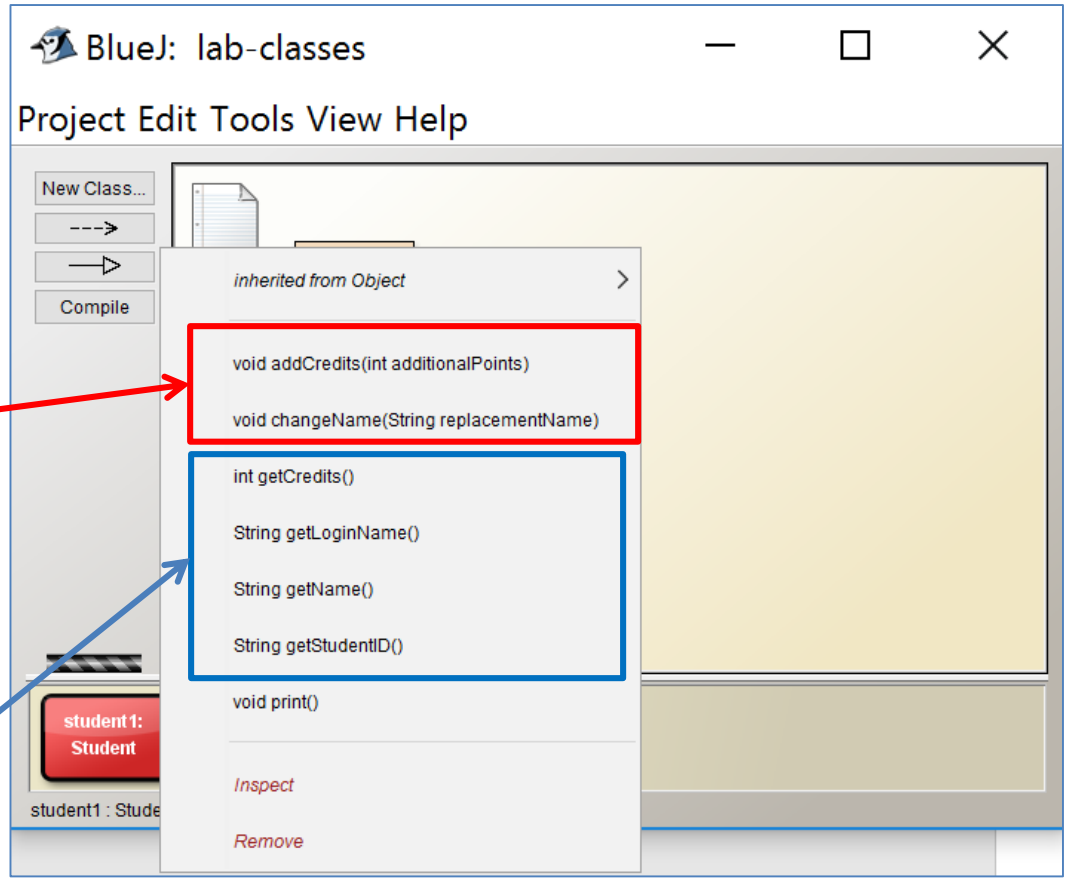


Return types

- Methods can return information about an object via a return value.

The void just before the method name means that nothing is returned from this method.

The int and String before the method names mean that something is returned from the method.



Return types (void)

```
void addCredits(int additionalPoints)
void changeName(String replacementName)
```

- The return type of these methods is void.
- These methods do not return any information.
- void is a return type and must be included in the method signature if your method returns no information.

Return types (when data is returned)

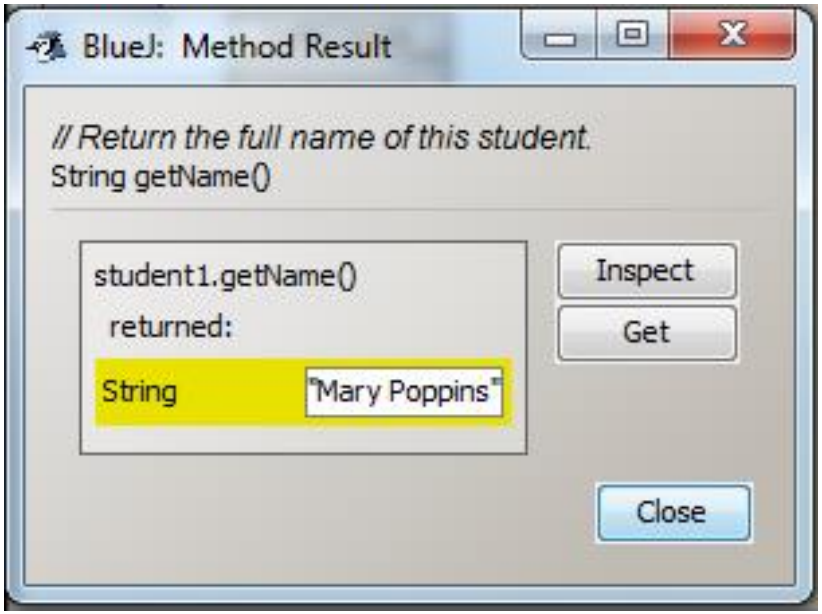
```
int getCredits()
String getLoginName()
String getName()
String getStudentID()
```

- Each of the above methods returns data.
  - The getCredits() method returns data whose type is int.
  - The getName() method returns data whose type is String.

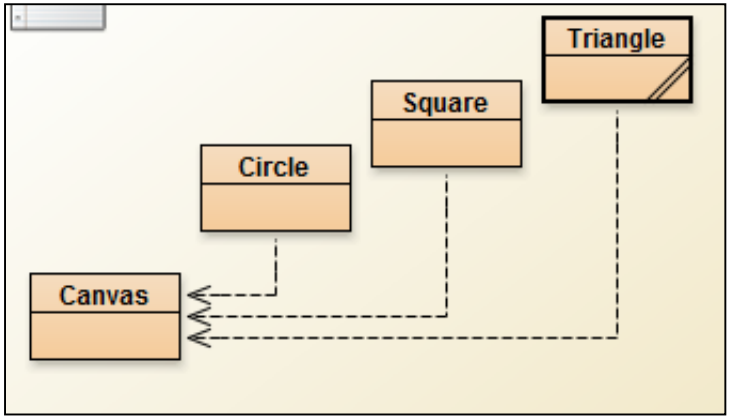
- You can only have one return type per method.

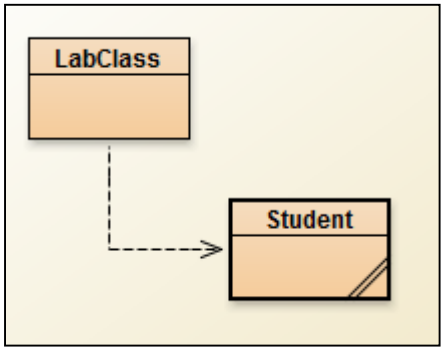
Return types (when data is returned)

- In BlueJ, when you call a method that returns data, a screen will pop up with the returned data e.g. the getName() method returns:



Naming conventions for Java classes





- All classes should start with a capital letter.
- Classes should be meaningfully named.
- Classes should be singular not plural.

Questions?

Study aid: Can you answer these questions?

- What is the purpose of parameters in Constructors?
- What are visibility/access modifiers?
- What is meant by passing Objects as parameters?
- What is a method signature?
- What are method return types?
- What is the accepted naming convention for Java Classes?

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
http://www.wit.ie/