

Classes and Objects

An Introduction

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/>

Demo

shapes project
(including its folder/file structure)

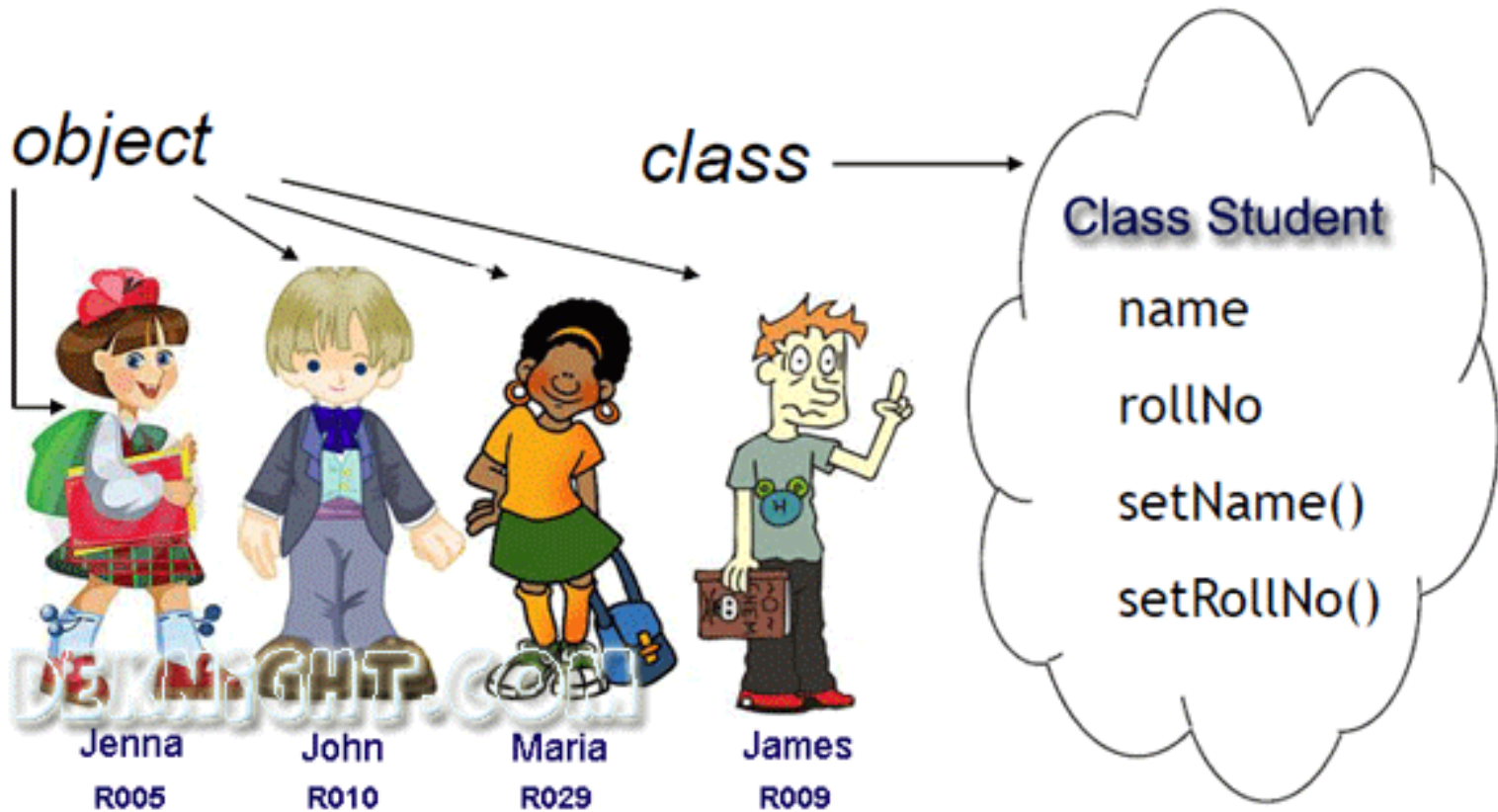
Topic List

- Understanding objects and classes
 - Classes
 - Objects and how to create them
 - Calling Methods
 - Parameters
 - Data Types
 - Multiple Instances
 - Object State
 - Object Interaction
- Files in Java
 - Source code
 - Compiling
- Java Virtual Machine

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

Student System: Objects and Classes



Returning to the *shapes* project
demonstrated earlier...

Objects

The screenshot shows the BlueJ IDE window titled "BlueJ: shapes". The main area displays a class hierarchy diagram with classes Canvas, Circle, Square, and Triangle. Dashed arrows indicate inheritance: Canvas is the superclass for Circle, Square, and Triangle. The bottom panel shows three object creation buttons: "circle 1: Circle", "square 1: Square", and "triangle 1: Triangle". A status bar at the bottom indicates "Creating object... Done."

BlueJ: shapes

Project Edit Tools View Help

New Class...

--->

->

Compile

Canvas

Circle

Square

Triangle

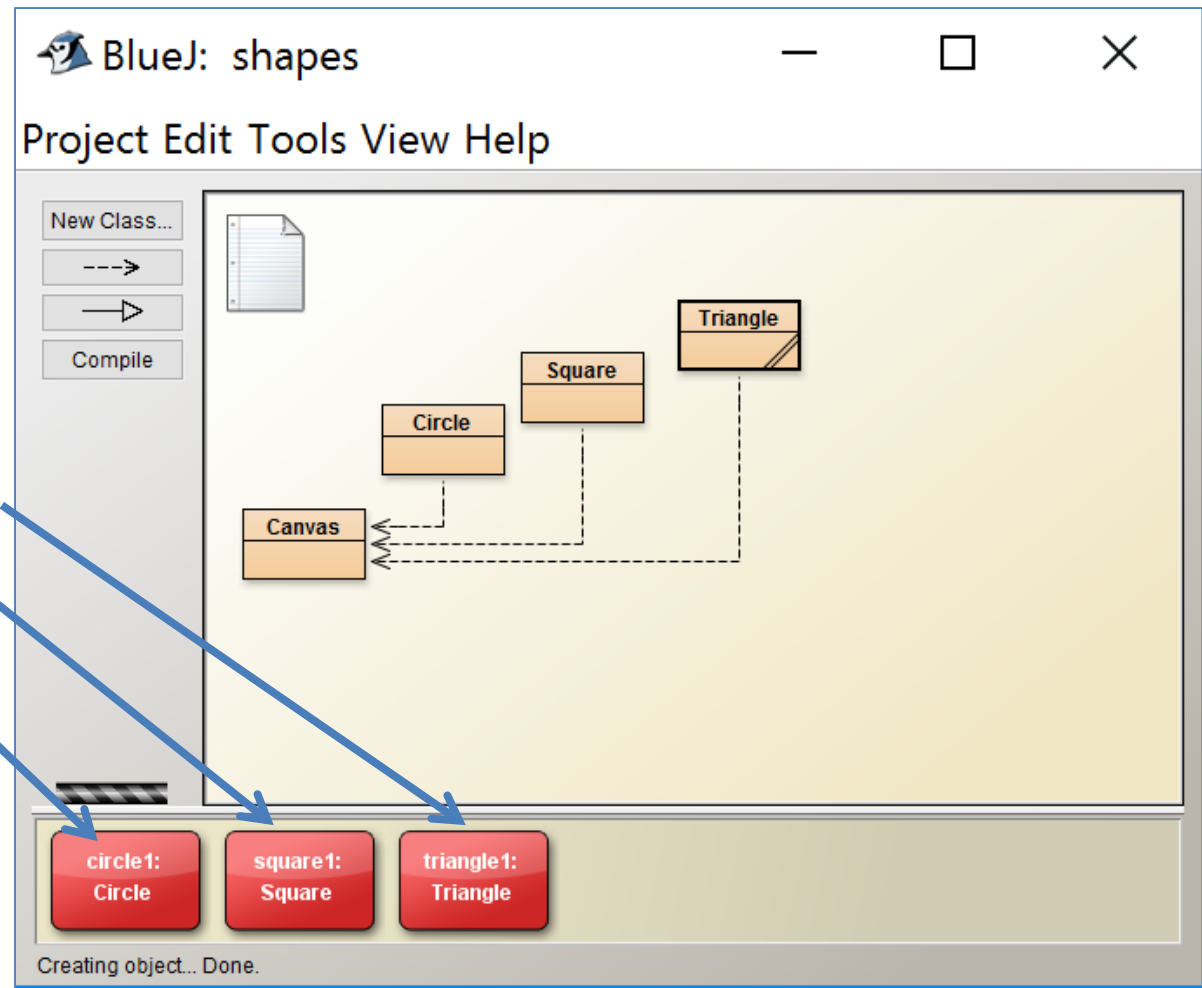
circle 1: Circle

square 1: Square

triangle 1: Triangle

Creating object... Done.

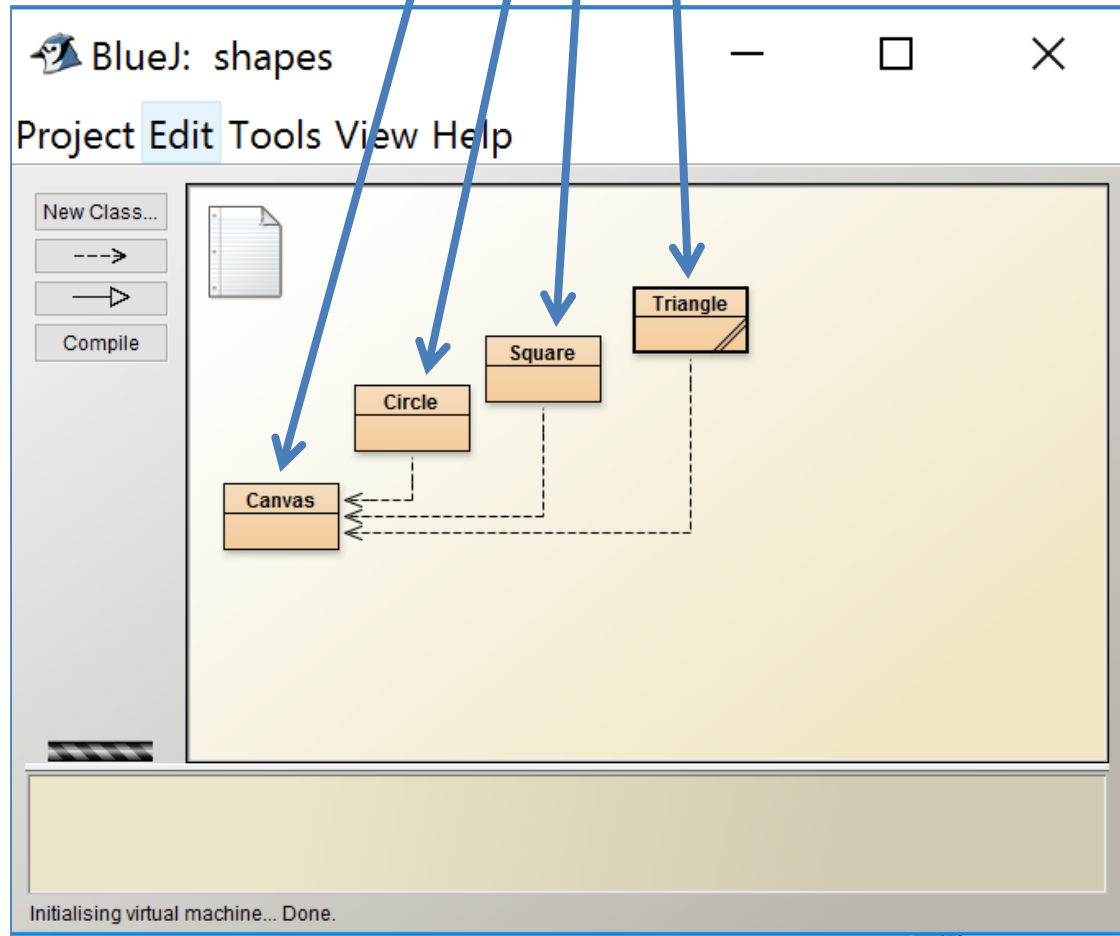
Objects



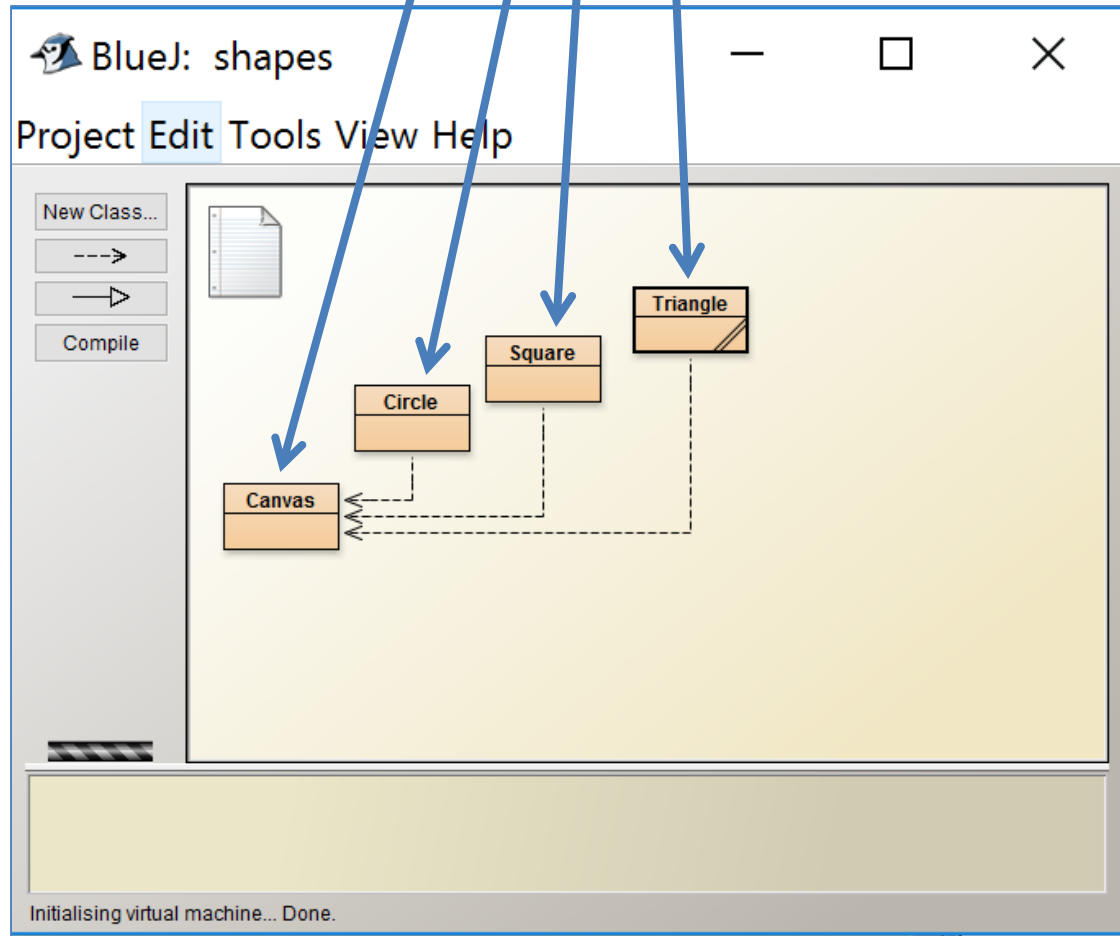
Represent 'things' from the real world, or from some problem domain e.g.:

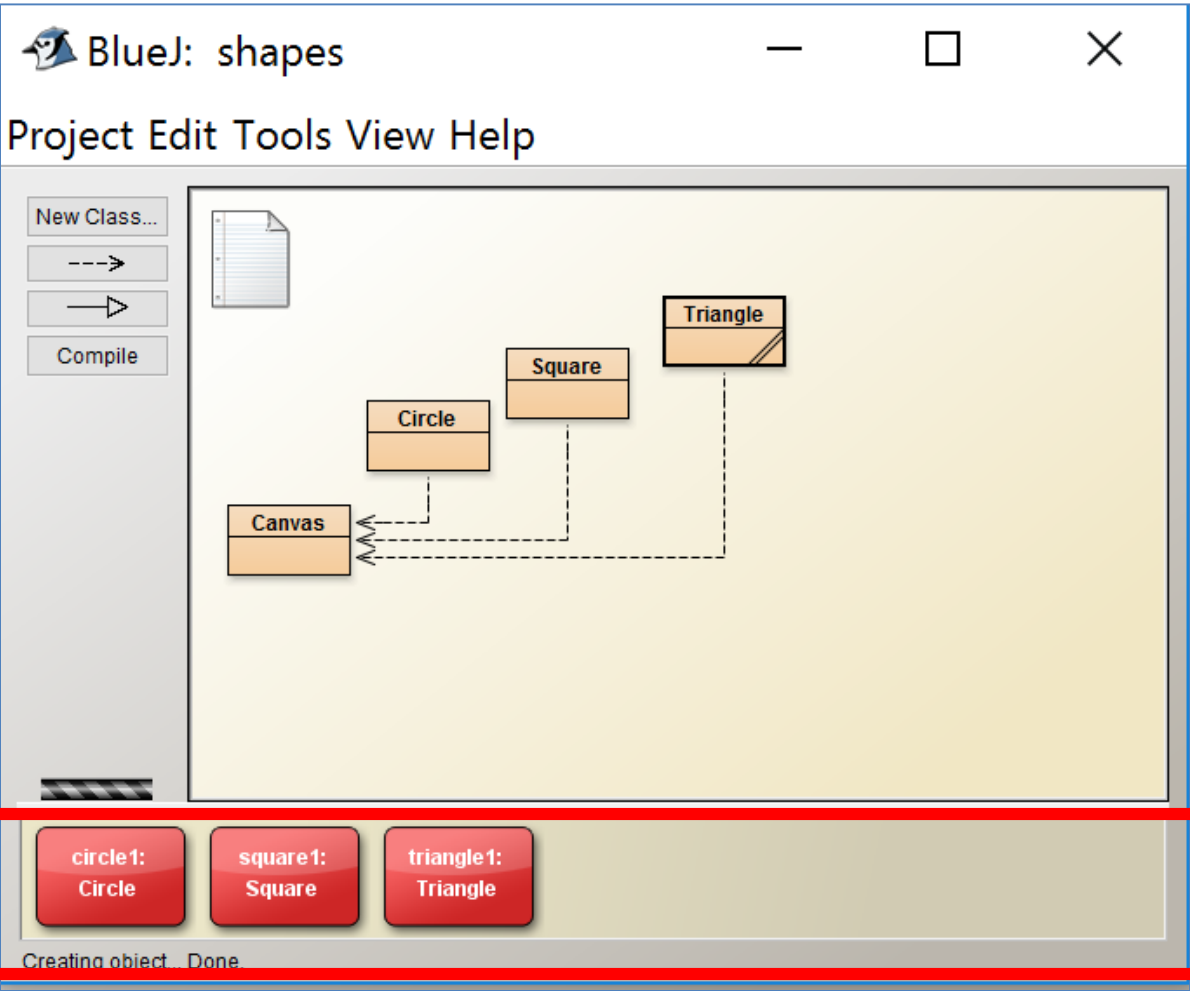
“the red car in the car park”

Classes



Classes





Object bench

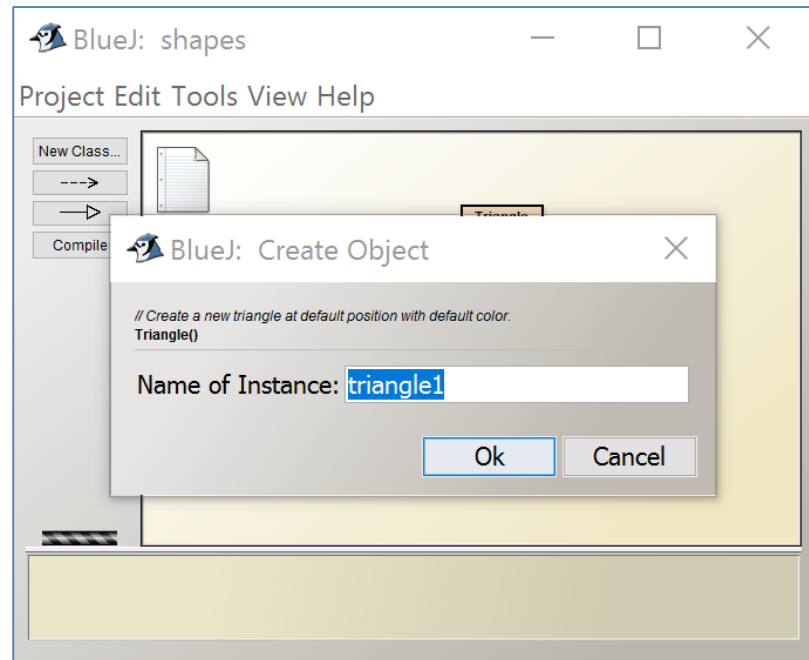
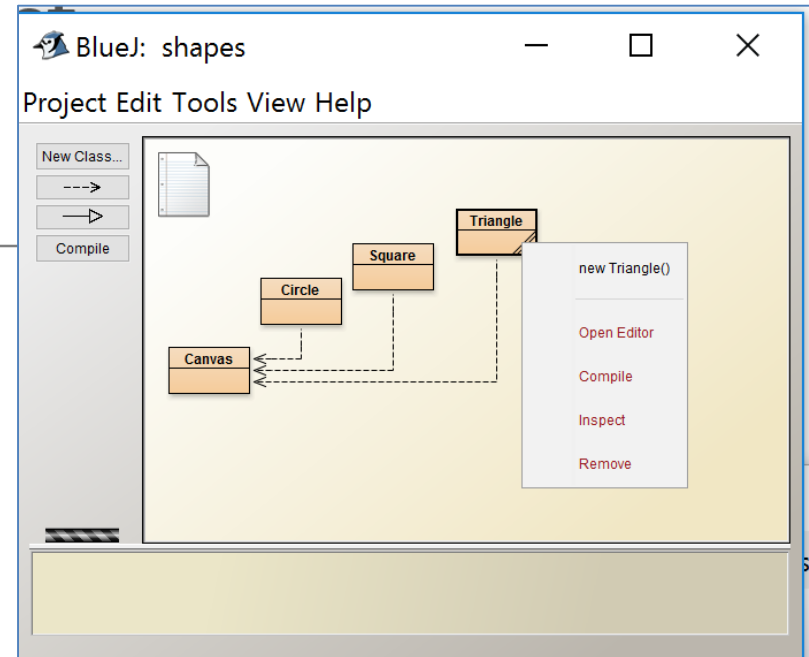


Objects and Classes

- Java objects model objects from a problem domain.
- Objects are created from classes.
- The class describes the kind of object; the class is a **template/blueprint**.
- The objects represent individual **instantiations** of the class.
- An object is an **instance** of a class.

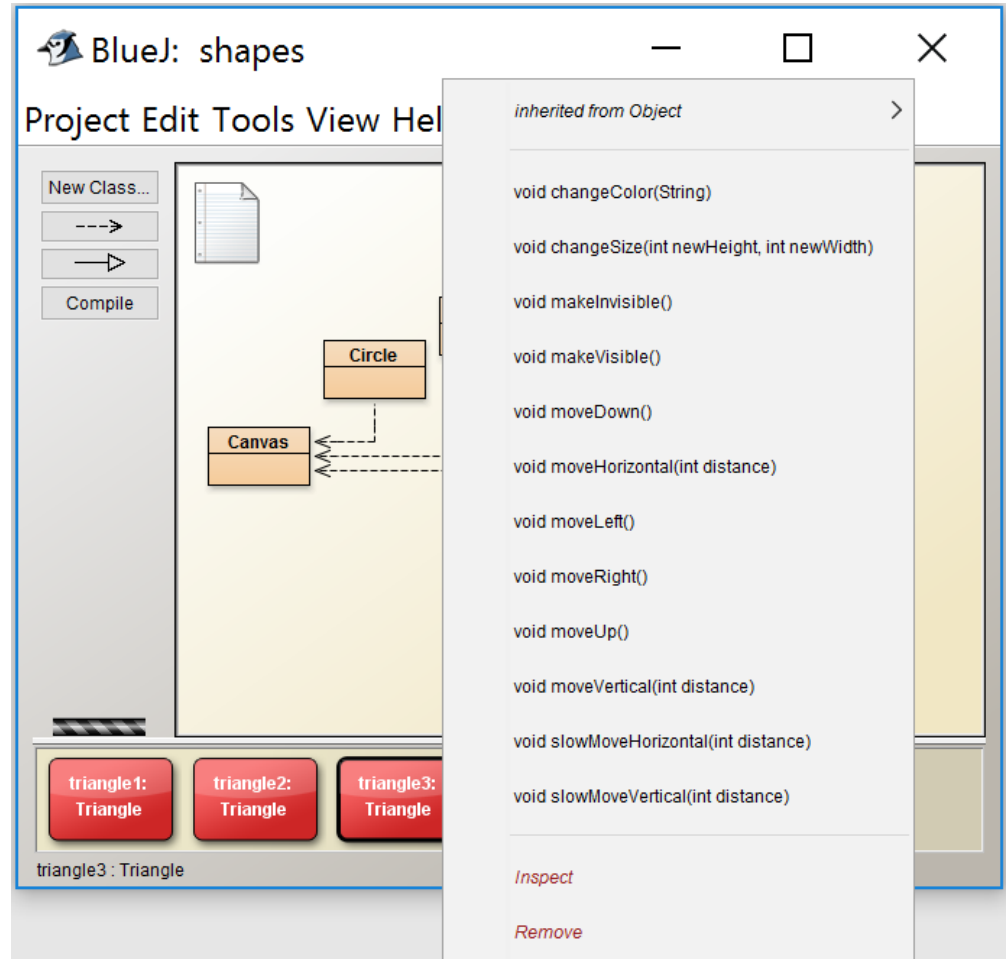
Creating an object

- 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.



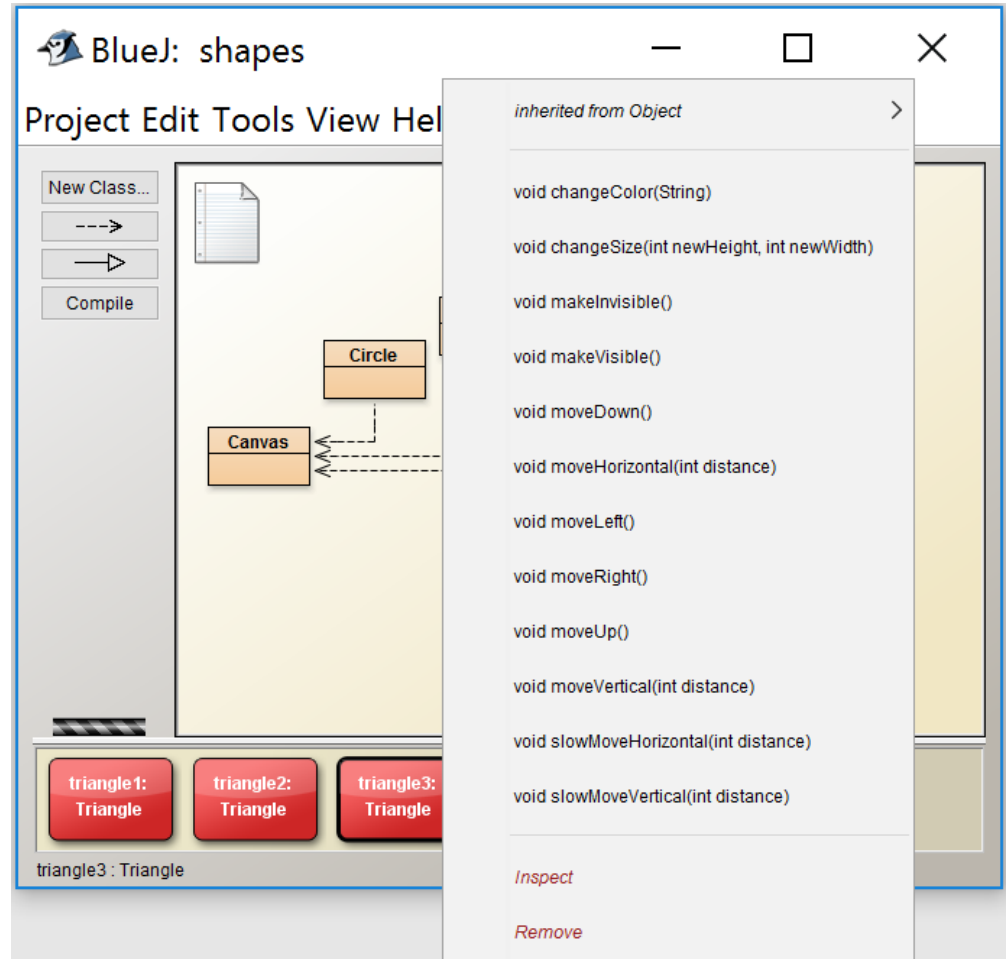
Methods

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



Calling methods (invoking)

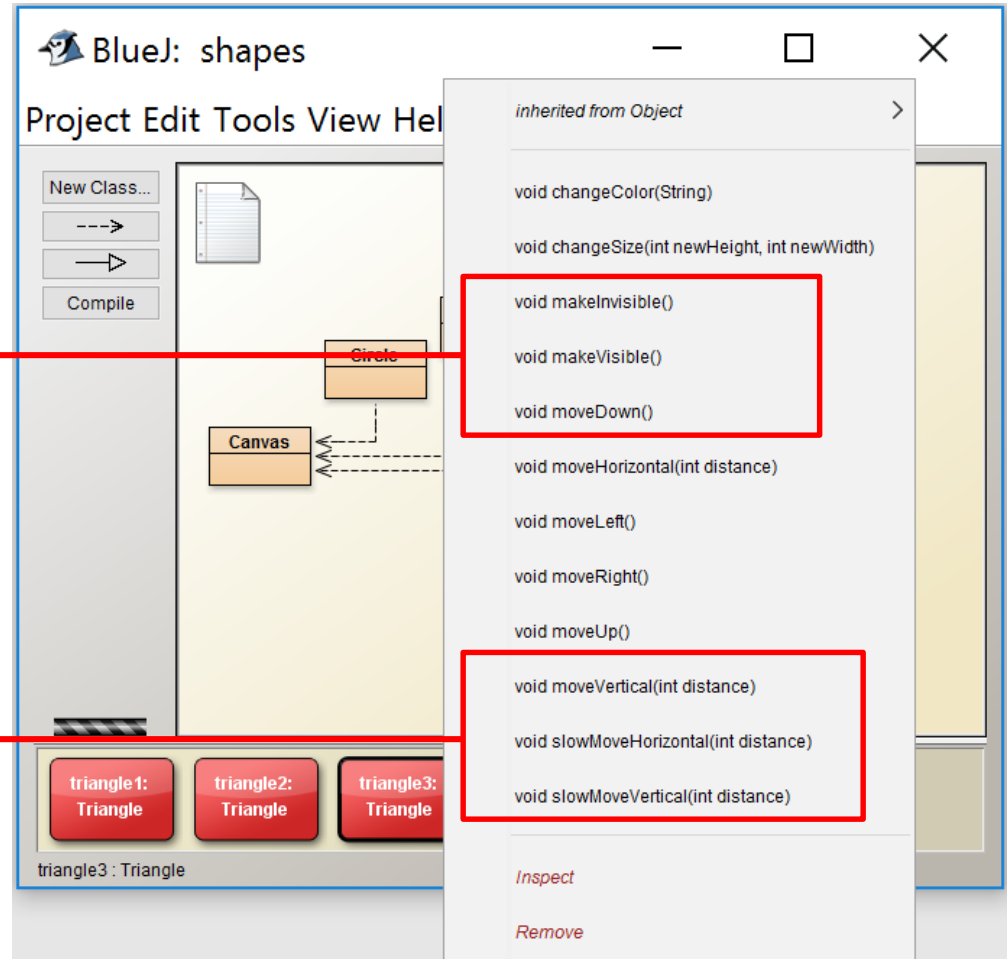
- Right click on the **object**.
- The popup menu lists all the methods that can be invoked on the object.
- Objects usually do something if we invoke a method.
- We can communicate with objects by invoking methods on them.



Parameters

These methods have
NO parameters

These methods
HAVE parameters



Methods with NO parameters

```
void makeInvisible()
```

```
void makeVisible()
```

```
void moveDown()
```

- Parameters to pass additional information needed to execute. Methods do not have to pass parameters.
- These methods have no parameters; note how no variable is passed in the parenthesis i.e. ().
- These methods don't need any additional information to do its tasks.

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).

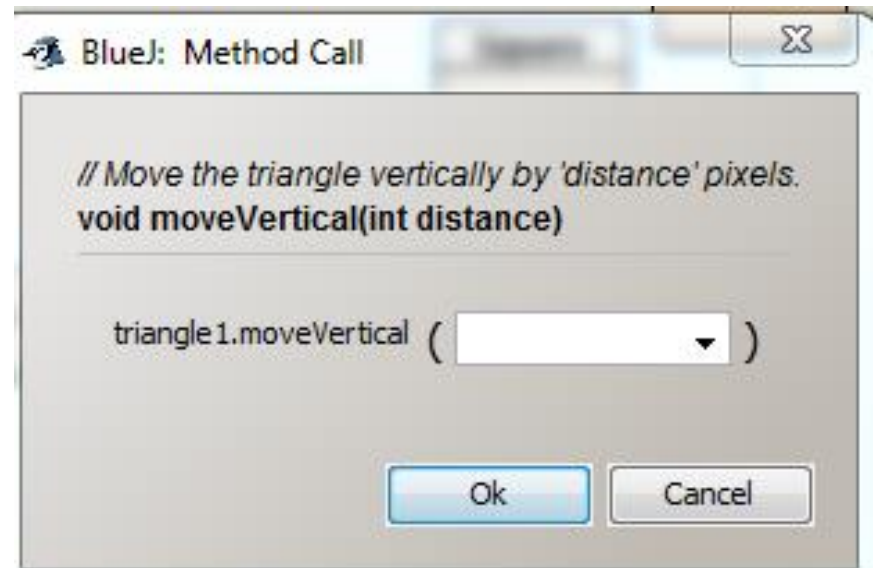
Methods with Parameters

```
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

- In programming, 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**
- A variable name must follow the conventions set out for it and we will cover these in a later lecture.

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.

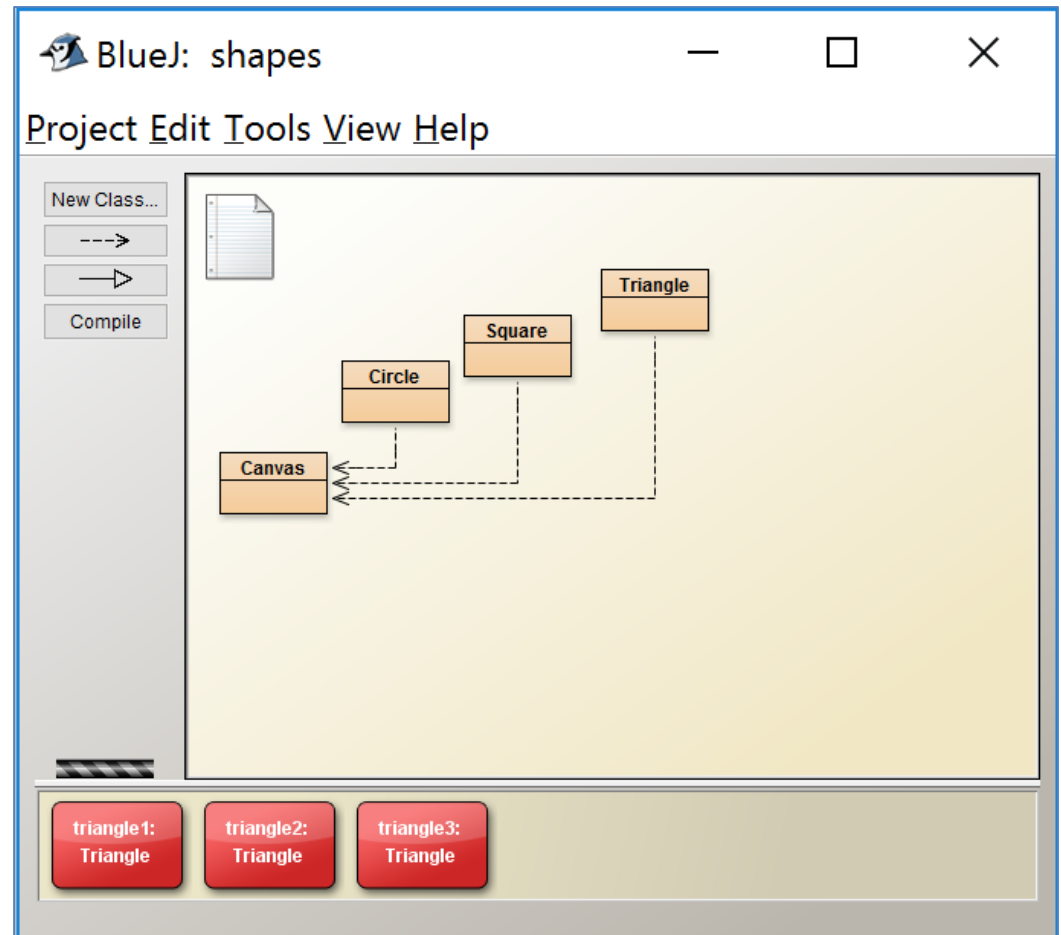
- **String**

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.

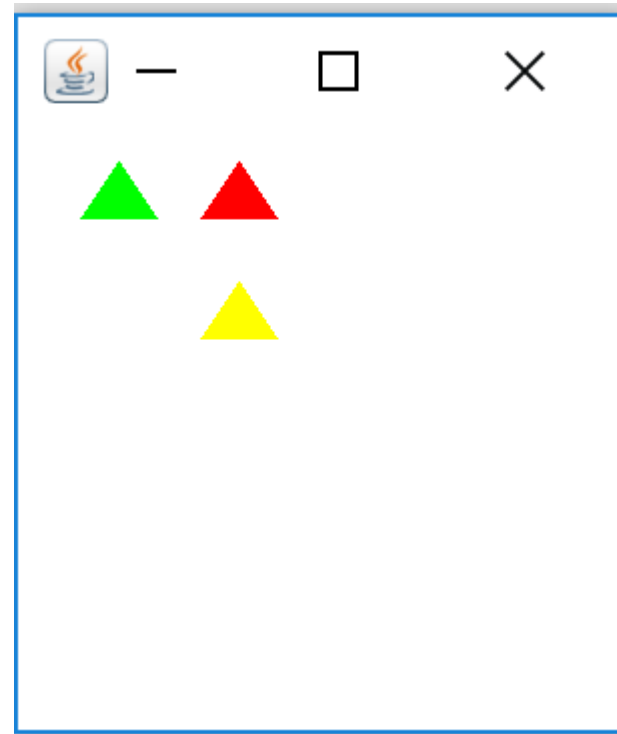
Multiple Instances

- 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.

triangle1 : Triangle

private int height	30	Inspect
private int width	40	Get
private int xPosition	50	
private int yPosition	15	
private String color	"green"	
private boolean isVisible	true	

Show static fields Close

triangle2 : Triangle

private int height	30	Inspect
private int width	40	Get
private int xPosition	110	
private int yPosition	15	
private String color	"red"	
private boolean isVisible	true	

Show static fields Close

triangle3 : Triangle

private int height	30	Inspect
private int width	40	Get
private int xPosition	110	
private int yPosition	75	
private String color	"yellow"	
private boolean isVisible	true	

Show static fields Close

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).

triangle1 : Triangle

private int height	30	Inspect Get
private int width	40	
private int xPosion	50	
private int yPosion	15	
private String color	"green"	
private boolean isVisible	true	

Show static fields Close

triangle2 : Triangle

private int height	30	Inspect Get
private int width	40	
private int xPosion	110	
private int yPosion	15	
private String color	"red"	
private boolean isVisible	true	

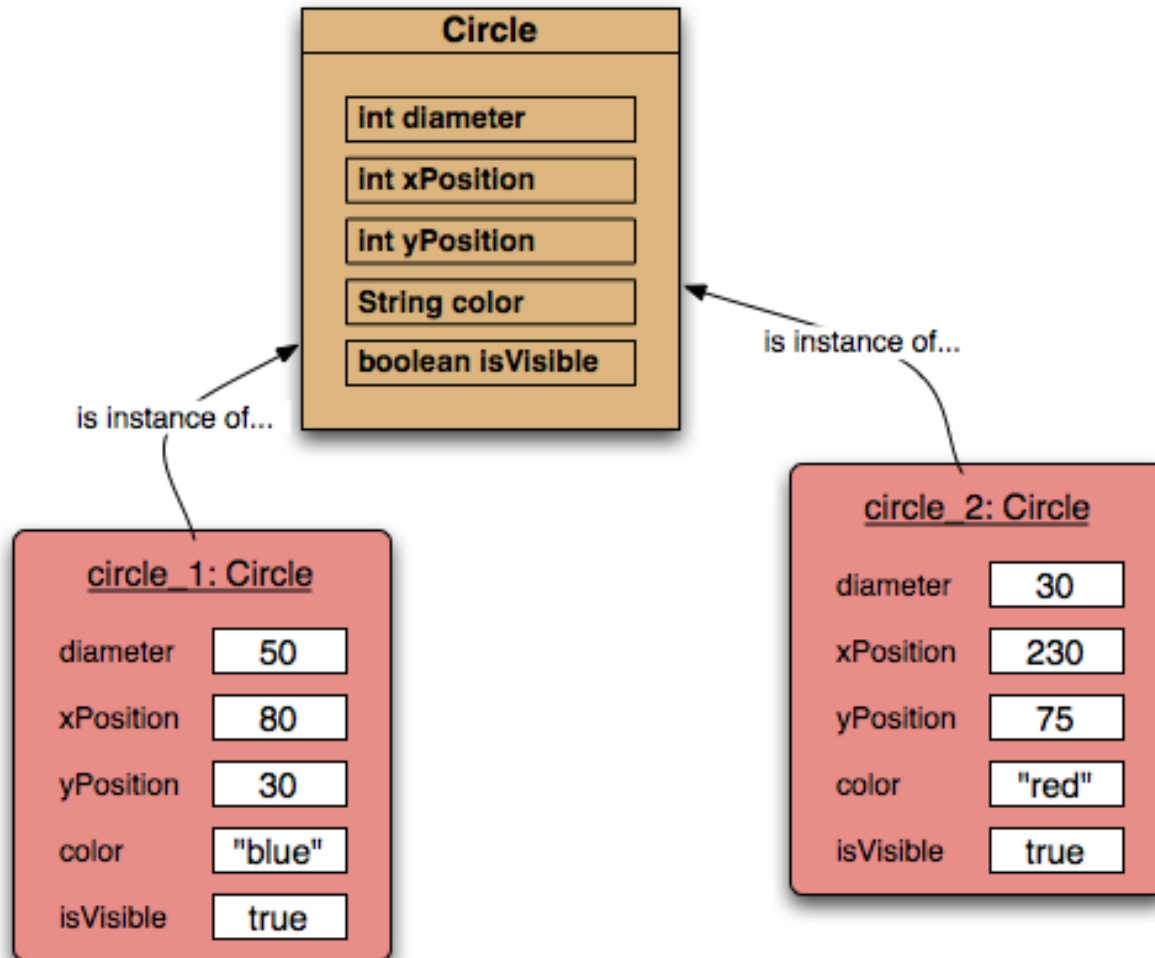
Show static fields Close

triangle3 : Triangle

private int height	30	Inspect Get
private int width	40	
private int xPosion	110	
private int yPosion	75	
private String color	"yellow"	
private boolean isVisible	true	

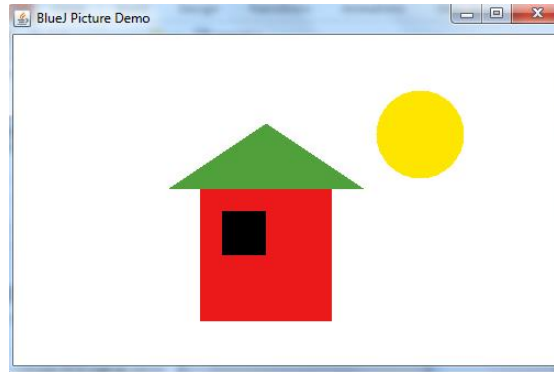
Show static fields Close

Another example: Two circle objects



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.

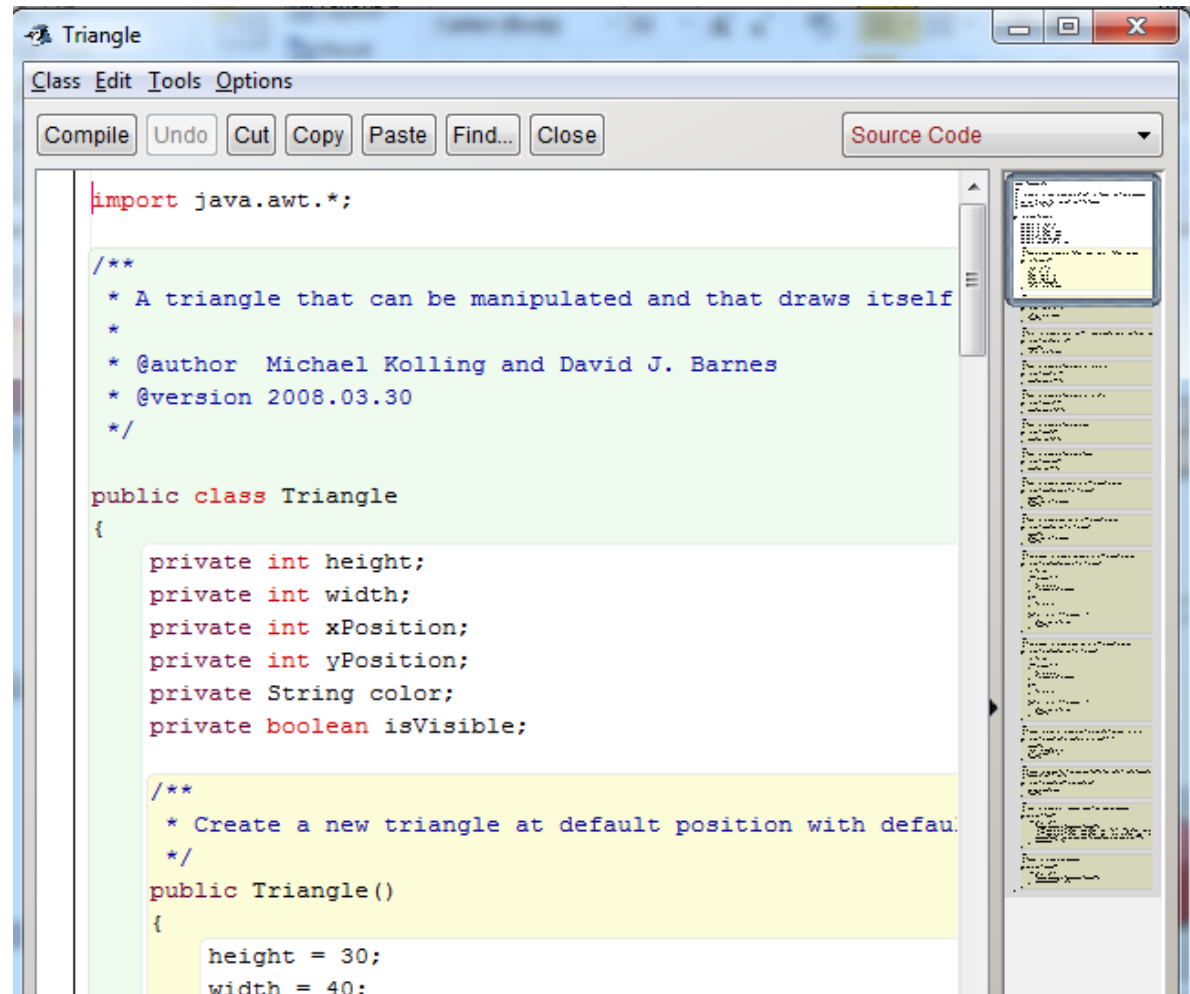
Topic List

- Understanding objects and classes
 - Classes
 - Objects and how to create them
 - Calling Methods
 - Parameters
 - Data Types
 - Multiple Instances
 - Object State
 - Object Interaction

- Files in Java
 - Source code
 - Compiling
- Java Virtual Machine

Source code

- Each class has Java source code.
- It defines the details of the class i.e. the variables and methods.
- It is stored in a .java file.
- The Triangle class is stored in a Triangle.java file.

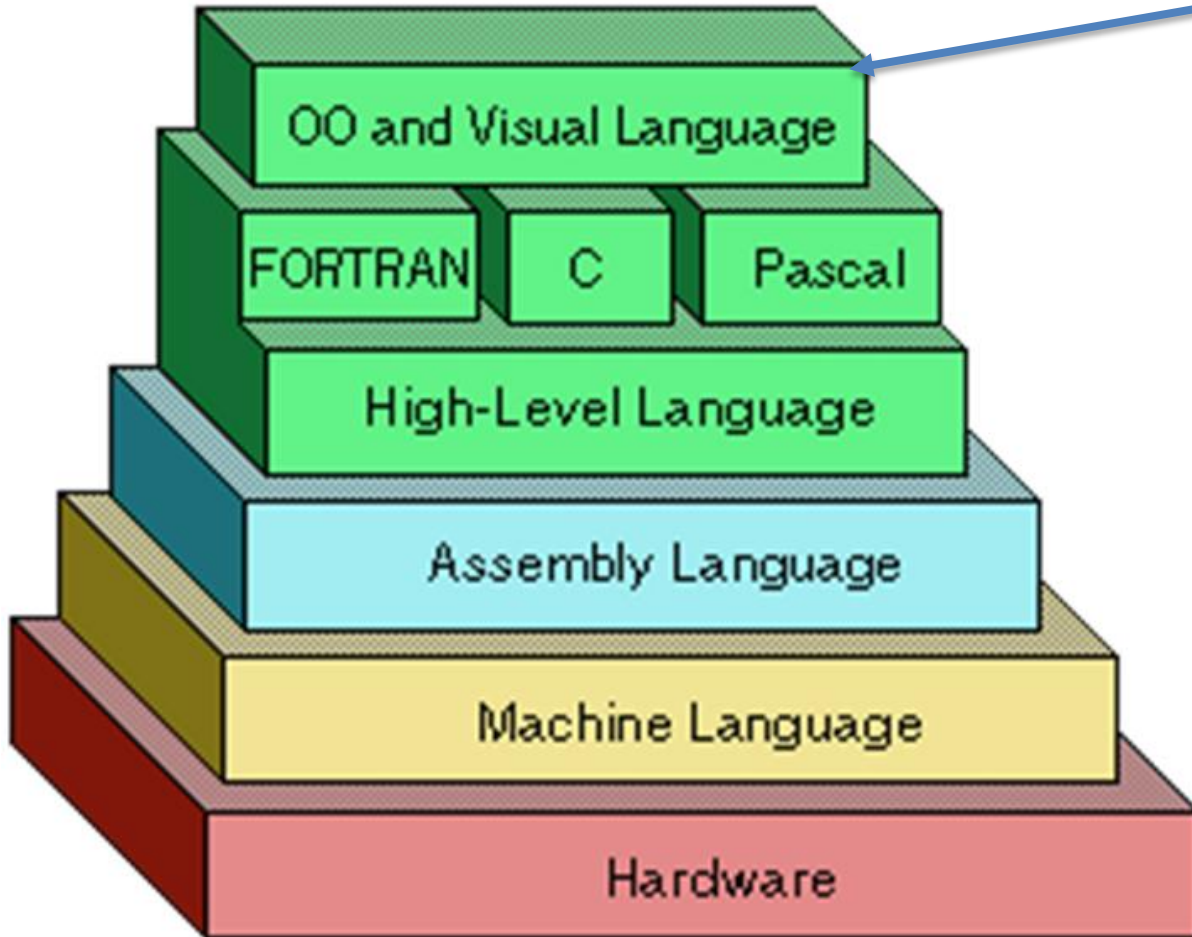


```
Triangle
Class Edit Tools Options
Compile Undo Cut Copy Paste Find... Close Source Code
import java.awt.*;

/**
 * A triangle that can be manipulated and that draws itself
 *
 * @author Michael Kolling and David J. Barnes
 * @version 2008.03.30
 */

public class Triangle
{
    private int height;
    private int width;
    private int xPosition;
    private int yPosition;
    private String color;
    private boolean isVisible;

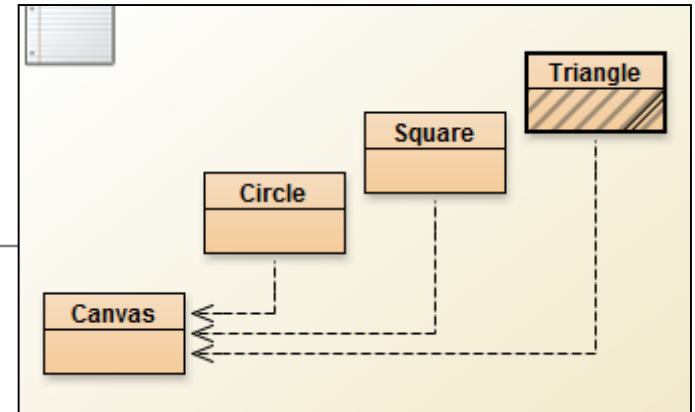
    /**
     * Create a new triangle at default position with default
     */
    public Triangle()
    {
        height = 30;
        width = 40;
```



This is where our java source code sits.

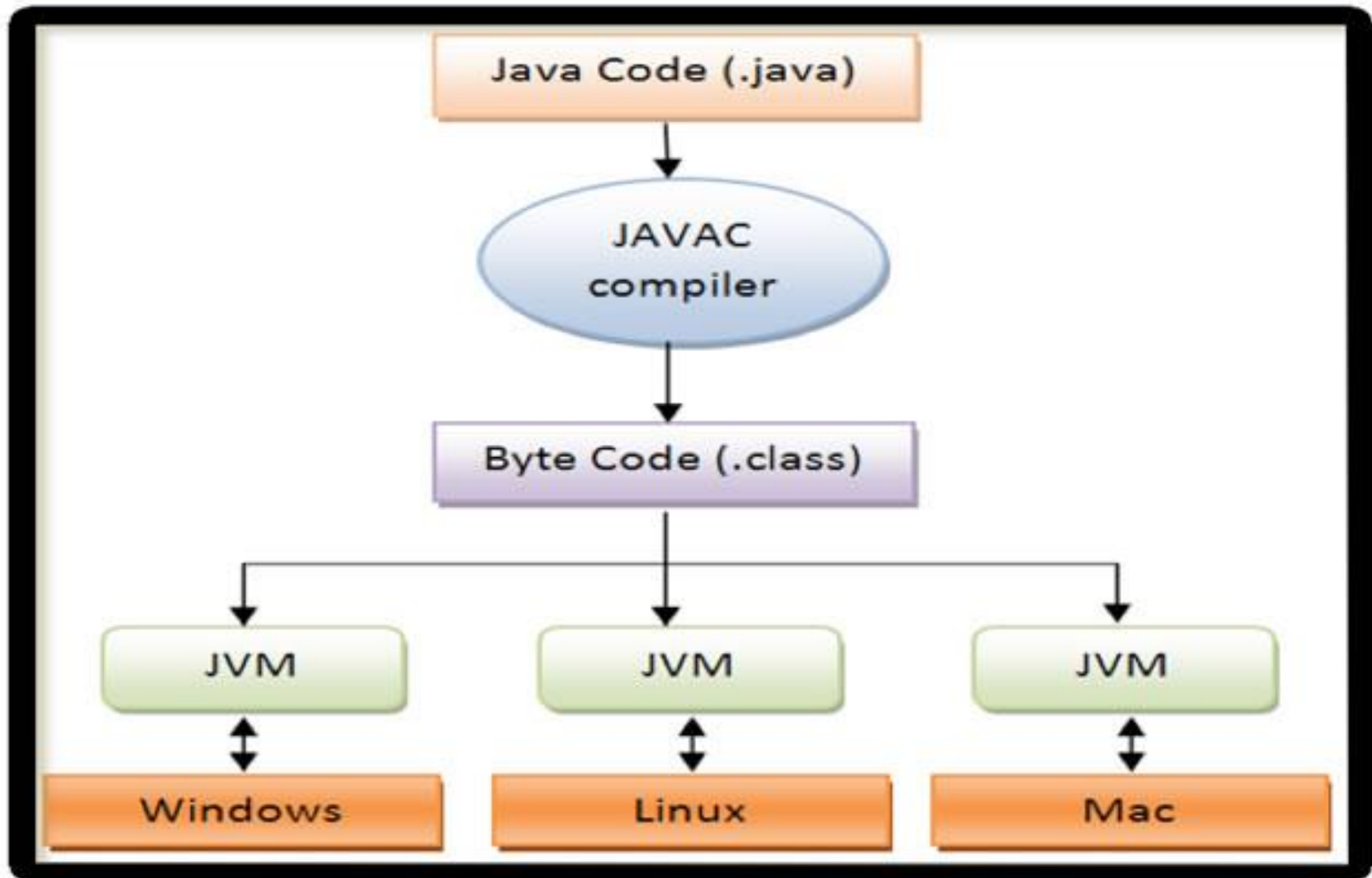
If we want it to run on a computer, we need to “convert” the source code into Machine Language that the specific hardware can understand i.e. compile it!

Compiling



- The compiler translates our Java code into machine language/code.
- When you make a change to the Source Code, you need to compile it.
- In the screen shot, the Triangle class needs to be compiled (see the stripes).
- When a class needs to be compiled, you cannot create an object from it. When you compile the class, you can create objects again.
- Compiling creates a .class file e.g. Triangle.class

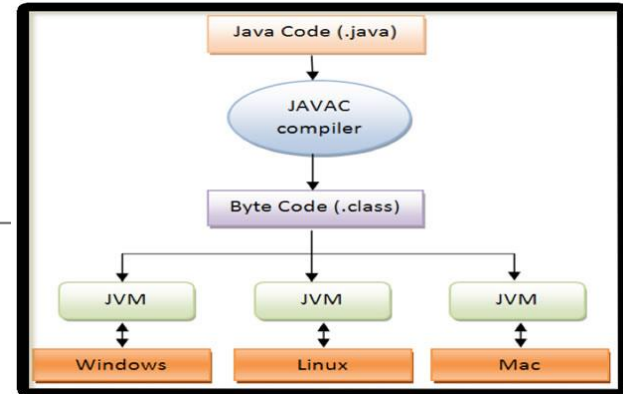
Compiling



Files in Java

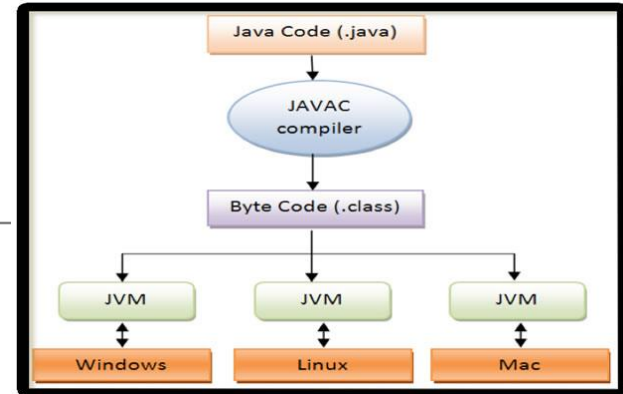
- Java code is written in **.java** file.
 - Each class has source code (Java code) associated with it that defines its details (fields and methods).
- When you compile this source code, a **.class** file is generated.
 - This source code is compiled into “**byte code**”.
 - It is the bytecode that you use to run the program.
 - The Java Virtual Machine (JVM) reads this code, interprets it and executes the program.

Java Virtual Machine



- Java Virtual Machine (JVM) is a “virtual” computer that resides in the “real” computer as a software process.
- When you install Java on your computer (JDK), you are also installing a Java Virtual Machine (JVM).

Java Virtual Machine



- It's job is to convert the bytecode into machine language that your hardware understands.
- Most programming languages compile directly into machine language, but Java compiles to JVM level. So the generated bytecode (.class files) can be run on ANY operating system that has a JVM installed i.e. platform-independence.

Questions?



Study aid: Can you answer these questions?

- What are classes?
- What are objects?
- How do we create objects?
- What are methods?
- How do we call them?
- What are parameters?
- What are data types?
- What is meant by Multiple Instances?
- What is object state?
- How do objects interact?

Study aid: Can you answer these questions?

- What is source code?
- What is compiling?
- What is byte-code?
- What is the JVM?
- What is a .java file?
- What is a .class file?



Except where otherwise noted, this content is licensed under a Creative Commons Attribution-NonCommercial 3.0 License.

For more information, please see <http://creativecommons.org/licenses/by-nc/3.0/>



Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

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