# 1.1: Objects and Classes

msklug.weebly.com

## Agenda:

- Attendance
- Intro
- Sign in
- Join Google
- Organize Google Drive
- Light Bot
- Creating Files
- Let’s try it out!

## Sign in!

## Google!

## LightBot

## Objects first?

- We are going to begin talking about programming within the language of Java
- We will start with the overall structure of Java code then dig in throughout the year

## What is Java?

- Java is a programming language
  - It is compiled into instructions that the computer can understand
  - It was developed by Sun Microsystems and is maintained by Oracle
  - Similar in syntax to C++ and C
  - We are using Java Development Kit (JDK) 1.6.0
  - This is an object-oriented programming language

## What is BlueJ?

- BlueJ is Integrated Development Environment (IDE)
  - It is a program that allows us to develop projects in Java
  - We can write, compile, run and debug in BlueJ

## What are Classes?

- Metaphorically, classes are the factories for the create objects.
- If you needed to create a circle, you would go to Circle class.
- Each circle you create is an object.
- Sometimes, we call each object we create an "instance" of the class.
- In BlueJ, a project is a collection of classes.

Convention: We start the names of classes with capital letters. We start instances with a lower case letter.

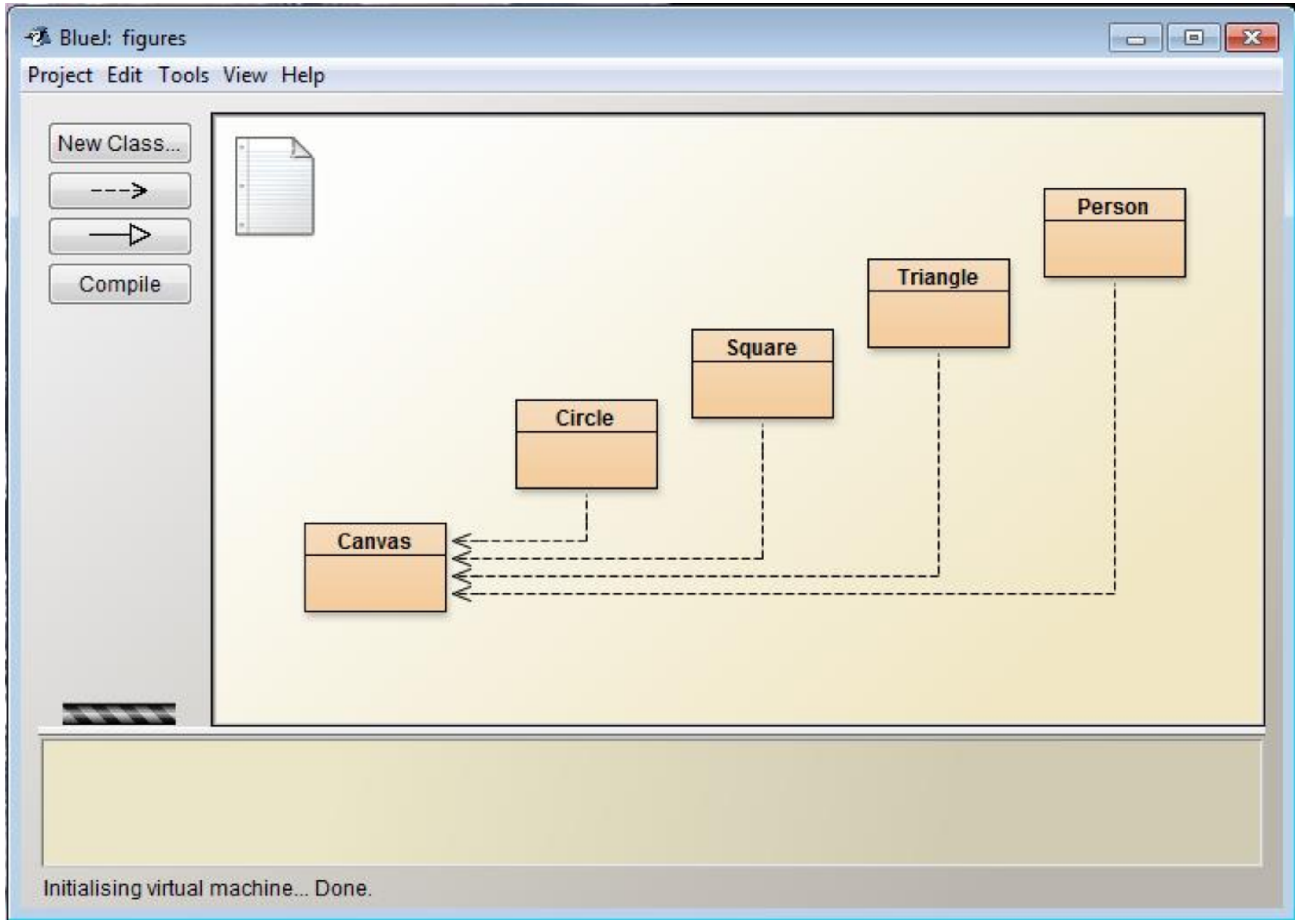
## Woah, wait what?

- Objects are created from classes, the class describes the *kind* of object
- Example:
  - Class: Human
  - Object/Instance: Ms. Klug

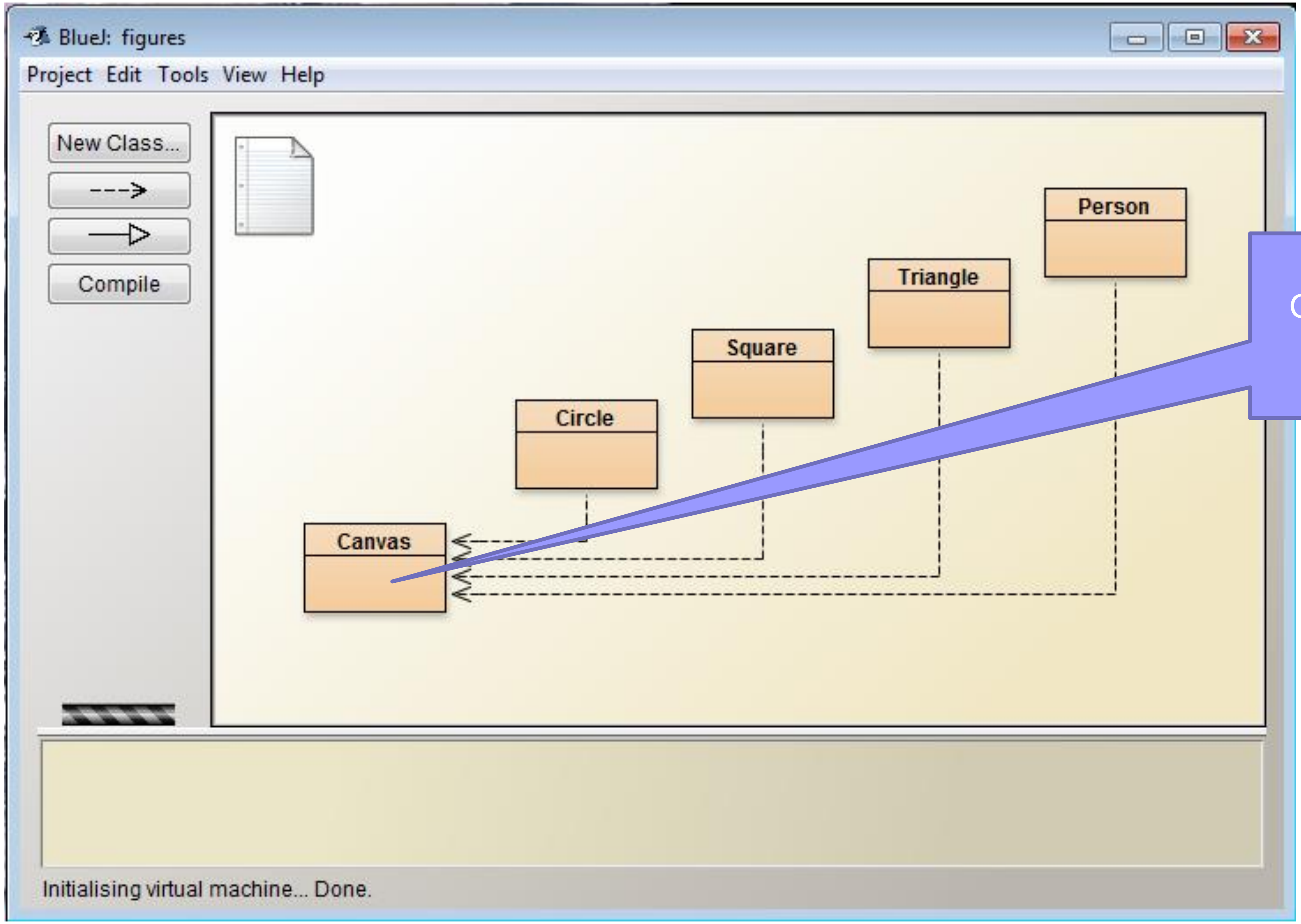
- Let’s come up with some other examples.

## Let’s get started

- Start BlueJ
- Open the project “figures” (on your desktop)
  - Make sure you SAVE AS
- Create a circle object

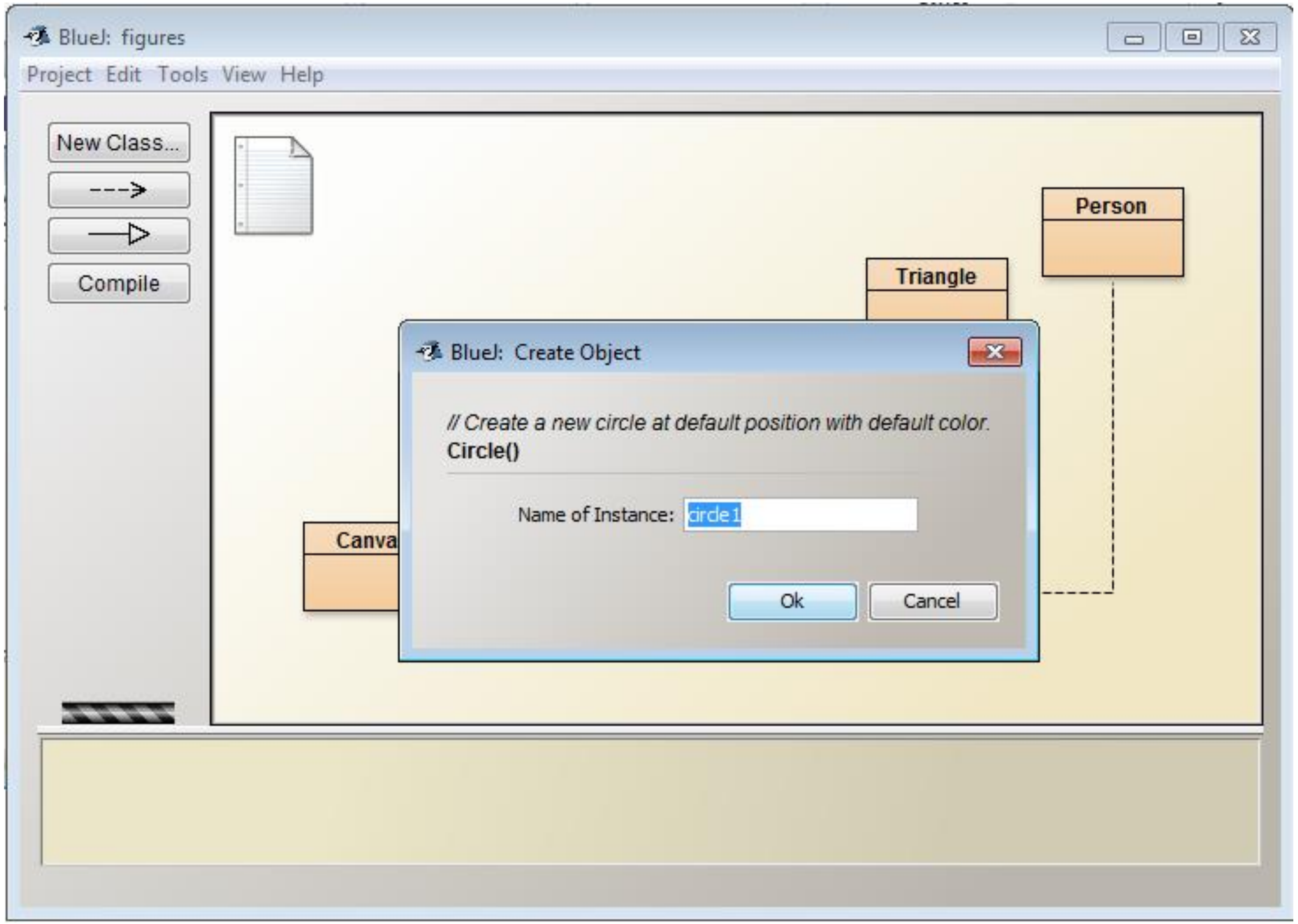


Initialising virtual machine... Done.



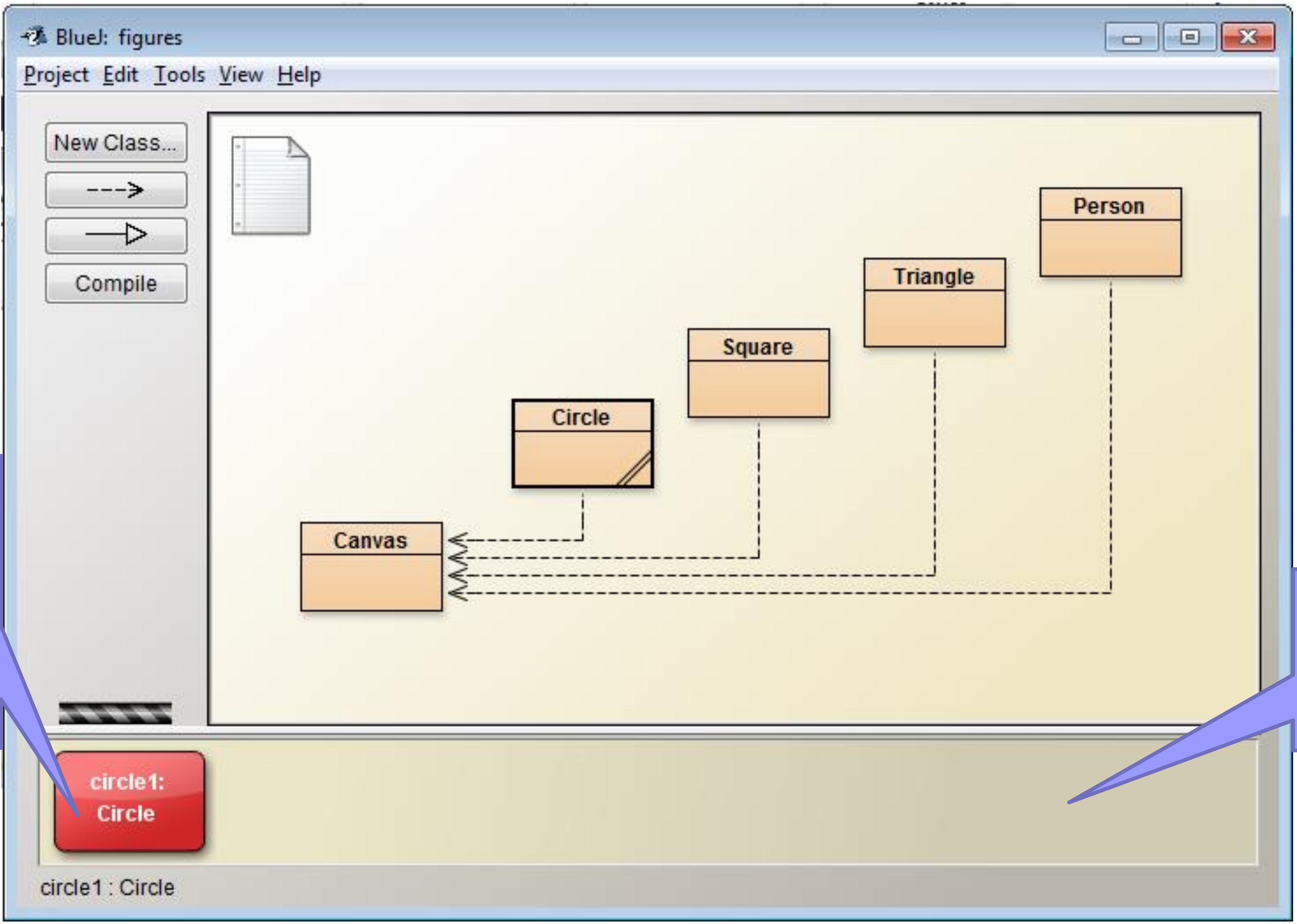
Initialising virtual machine... Done.

Class of type “Canvas”



// Create a new circle at default position with default color.
Circle()

Name of Instance: circle1



An instance of object Circle, (notice the lowercase first letter)

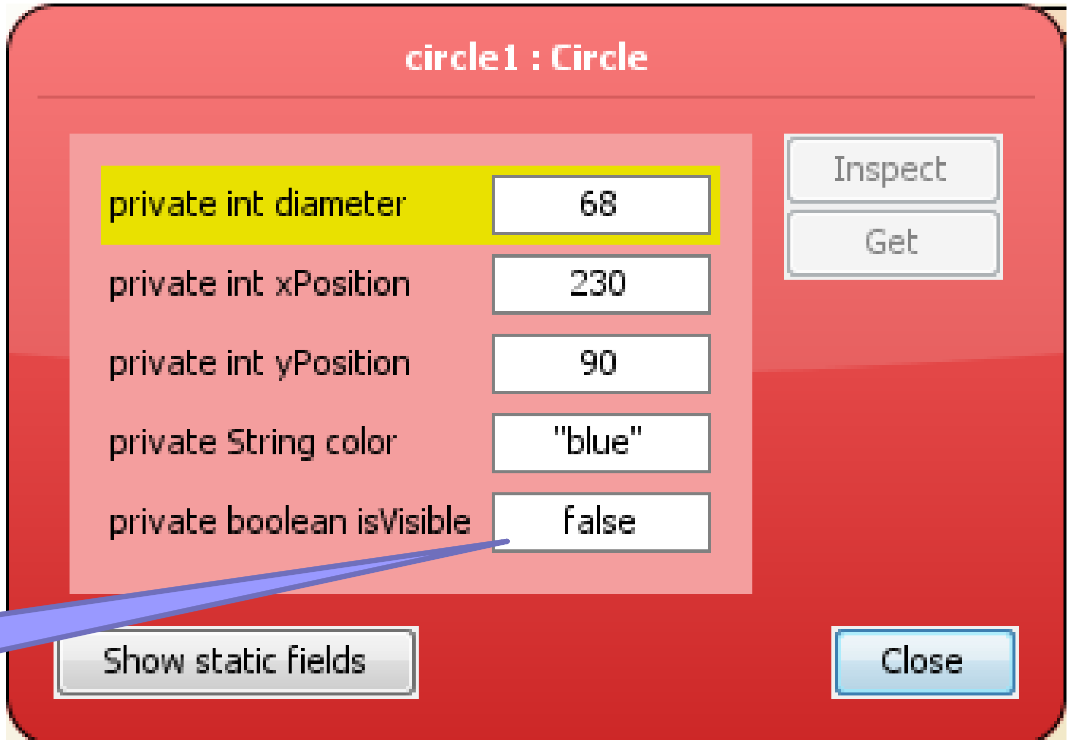
Object bench

## Exercise 1.1

- Go ahead!

## But I don't see a circle

- Double click on the instance of an object to see its attributes. Let's look at the attributes of the circle we created



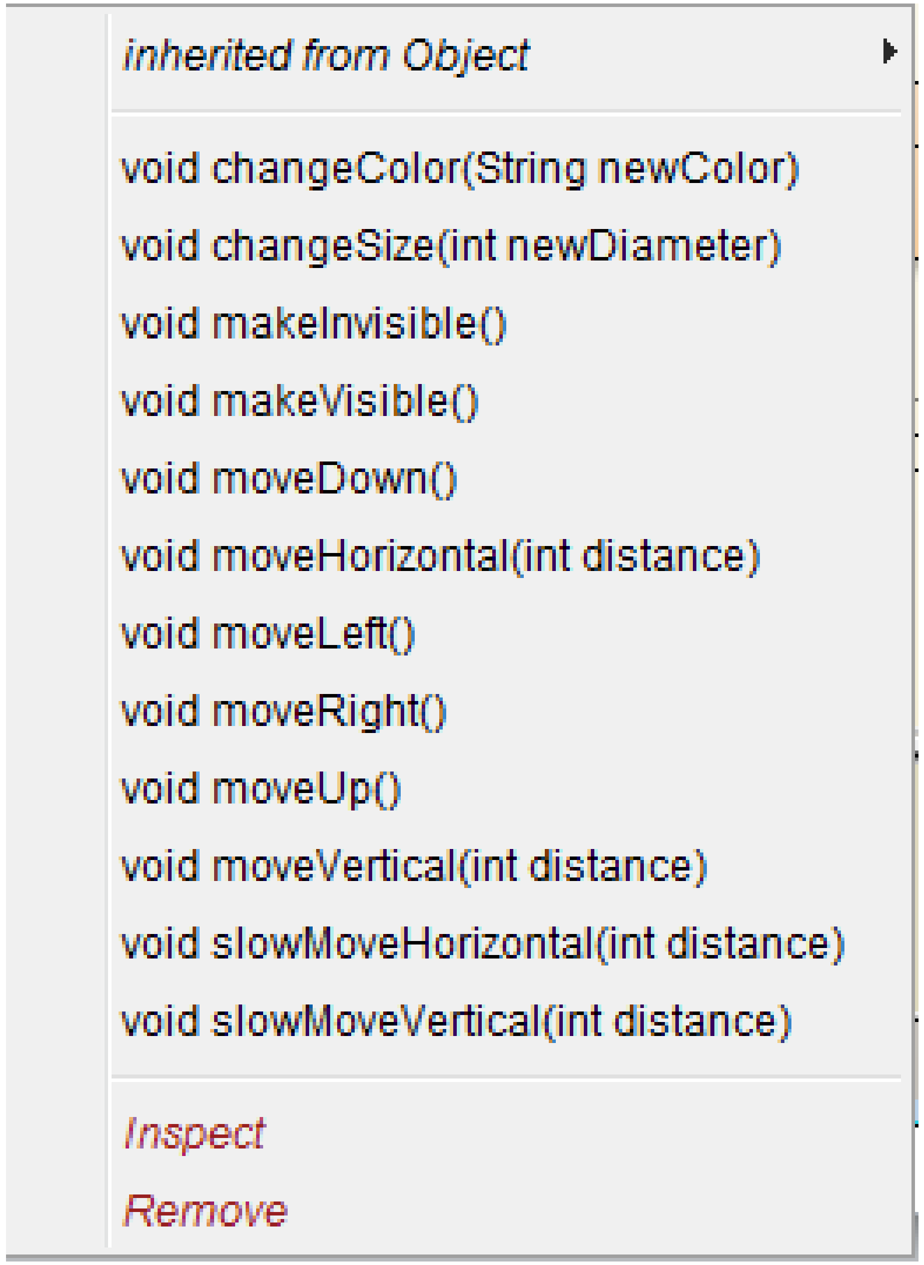
circle1 : Circle

| | |
|---|---|
| private int diameter | 68 |
| private int xPosition | 230 |
| private int yPosition | 90 |
| private String color | "blue" |
| private boolean isVisible | false |

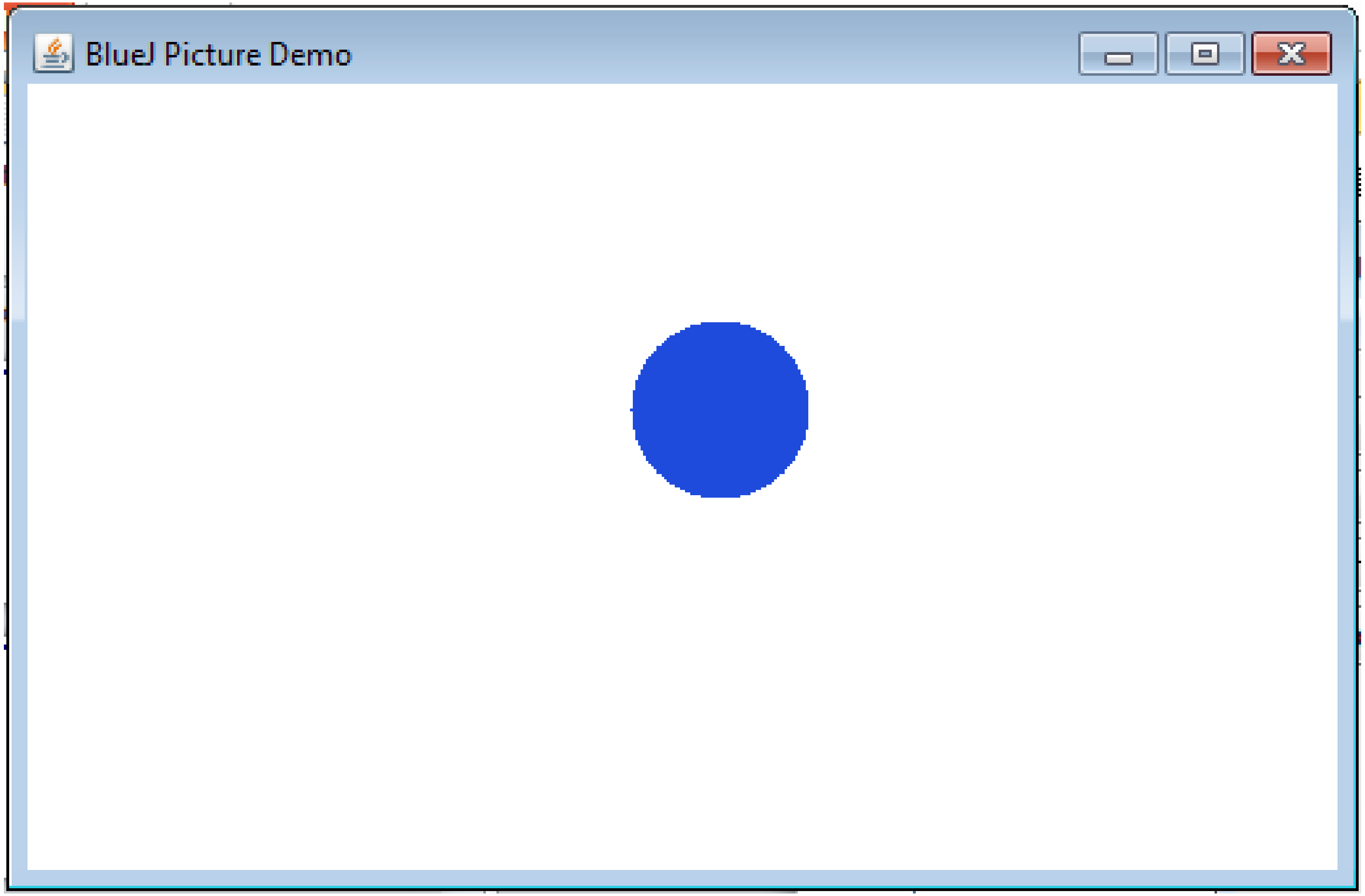
False, not visible

# Objects do things

- Object do things using methods
  - Let’s look at the methods associated with our circle object
- Let’s “call/invoke” makeVisible()



```
void changeColor(String newColor)
void changeSize(int newDiameter)
void makeInvisible()
void makeVisible()
void moveDown()
void moveHorizontal(int distance)
void moveLeft()
void moveRight()
void moveUp()
void moveVertical(int distance)
void slowMoveHorizontal(int distance)
void slowMoveVertical(int distance)
```

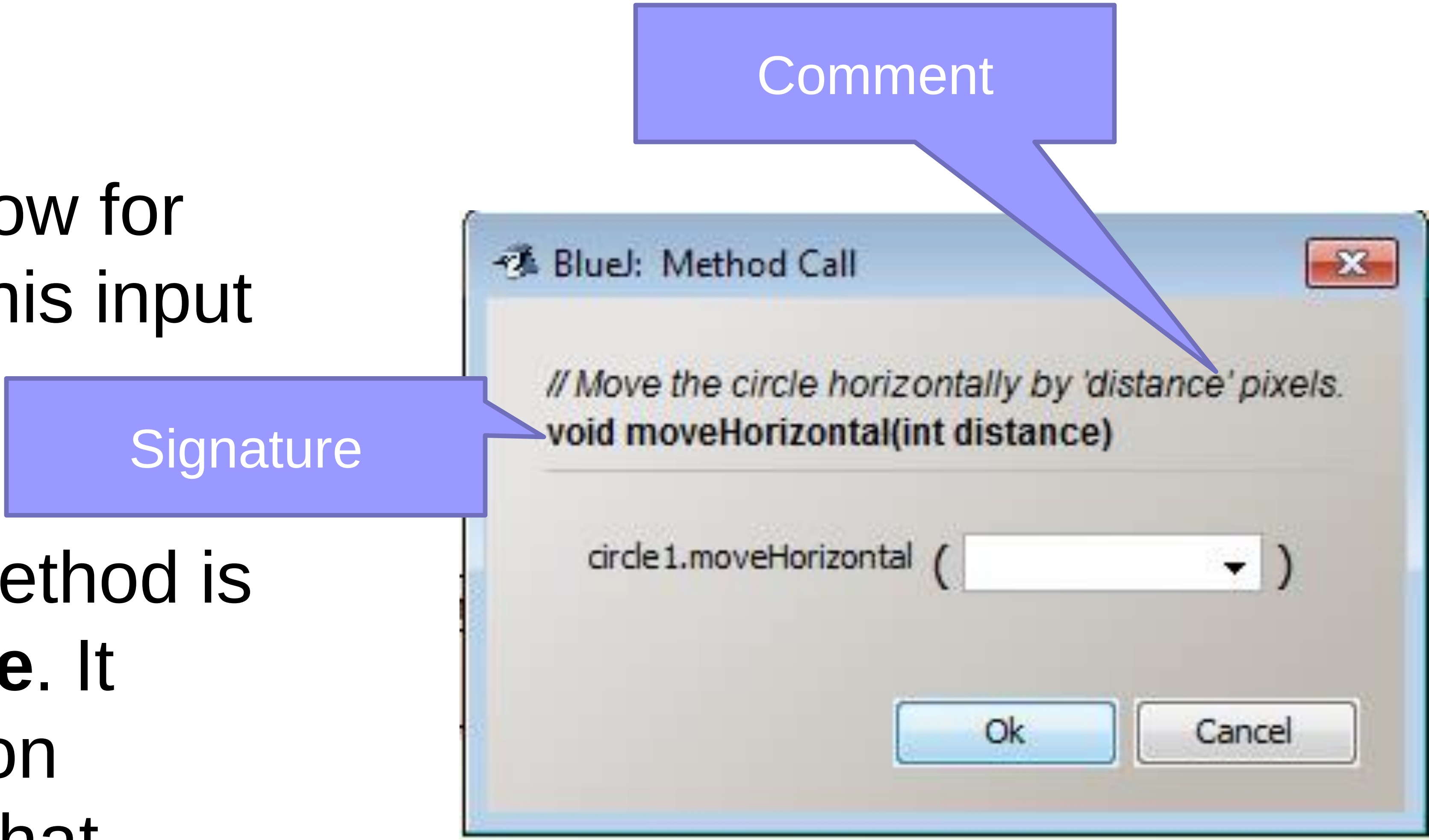

BlueJ Picture Demo

## Exercise 1.2

- You are now ready for this exercise

## Parameters

- Some methods allow for input of a value. This input values are called **parameters**.
- The header of a method is called its **signature**. It provides information needed to invoke that method.

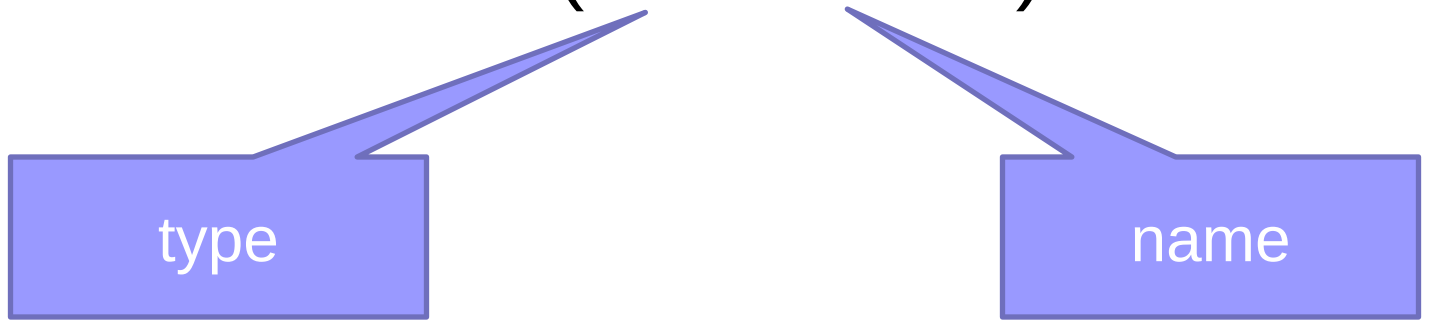


```
// Move the circle horizontally by 'distance' pixels.
void moveHorizontal(int distance)

circle1.moveHorizontal (  )
```

## More on the signature of a Method

- void moveHorizontal(int distance)



## Exercise 1.3

- You are now ready for this exercise.

## Review so far:

- BlueJ
- Java
- Object/Instance
- Class
- Method
- Parameters
- Signature

## Data types

- Parameters have **types**. The type defines what kinds of values a parameter can take.

- `int` signifies whole numbers.
- `String` indicates combinations of letters and symbols. Values for strings are always in double quotes.
- `boolean` indicates values that are either `true` or `false`.

## Exercises 1.4, 1.5, 1.6, and 1.7

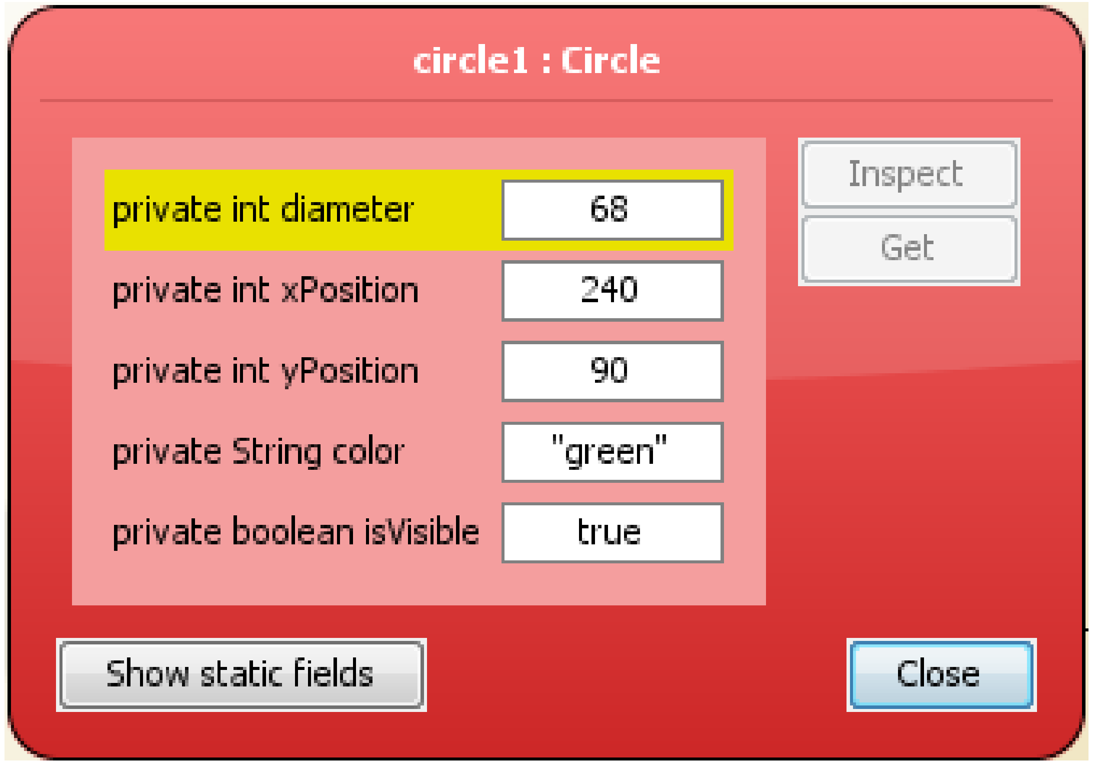
- Go for it!

## Multiple Instances

- Many similar objects can be created from a single class.

## State

- The values of all the attributes of an object is referred to as the object’s state in the object inspector.
- Java refers to these object attributes as **fields**.
- Double click on the instance of an object to see the object’s state.

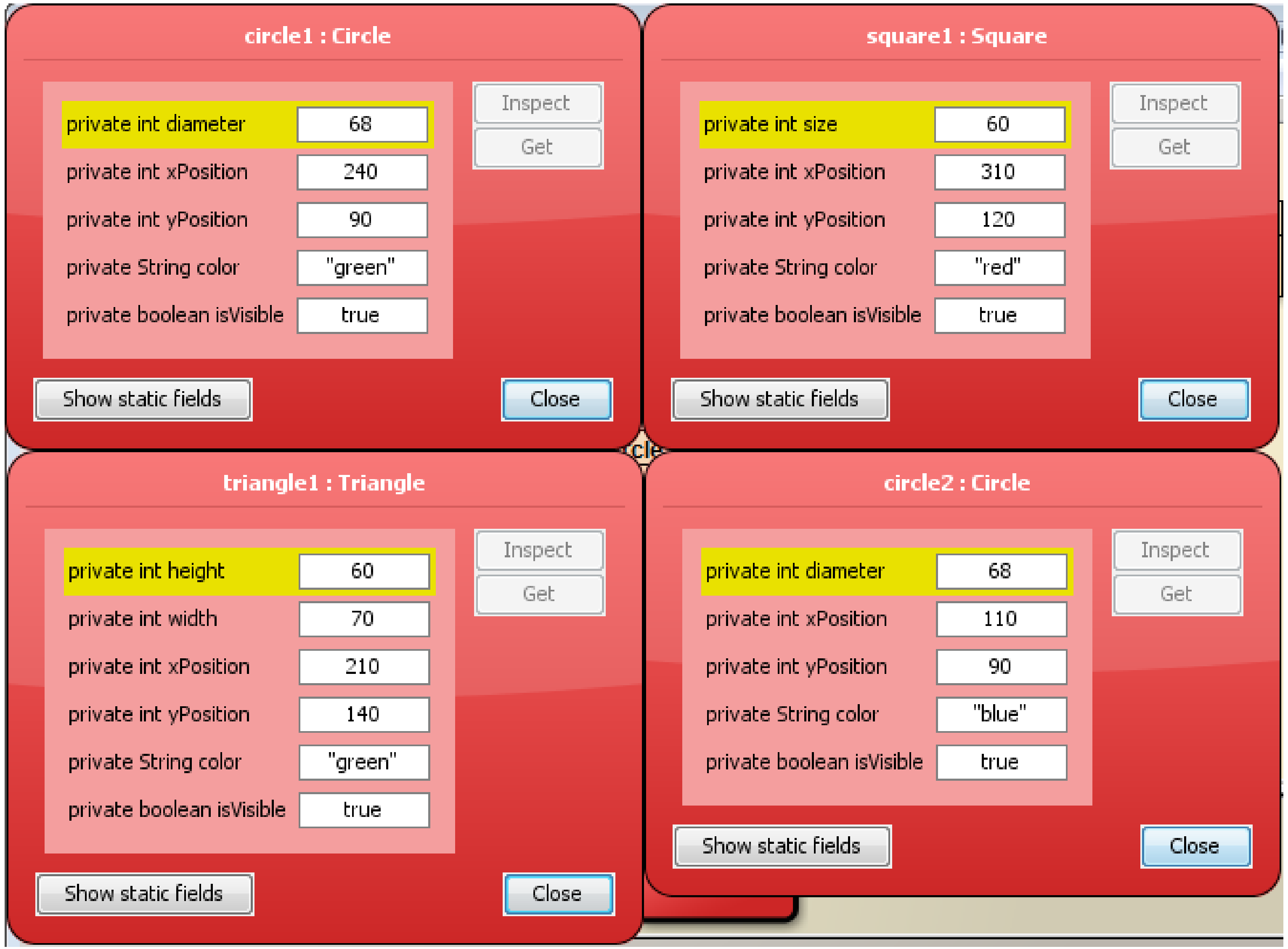

circle1 : Circle

| | |
|---|---|
| private int diameter | 68 |
| private int xPosition | 240 |
| private int yPosition | 90 |
| private String color | "green" |
| private boolean isVisible | true |

## Exercise 1.8

- It’s time!

# Different classes have different attributes



circle1 : Circle

| | |
|---|---|
| private int diameter | 68 |
| private int xPosition | 240 |
| private int yPosition | 90 |
| private String color | "green" |
| private boolean isVisible | true |

square1 : Square

| | |
|---|---|
| private int size | 60 |
| private int xPosition | 310 |
| private int yPosition | 120 |
| private String color | "red" |
| private boolean isVisible | true |

triangle1 : Triangle

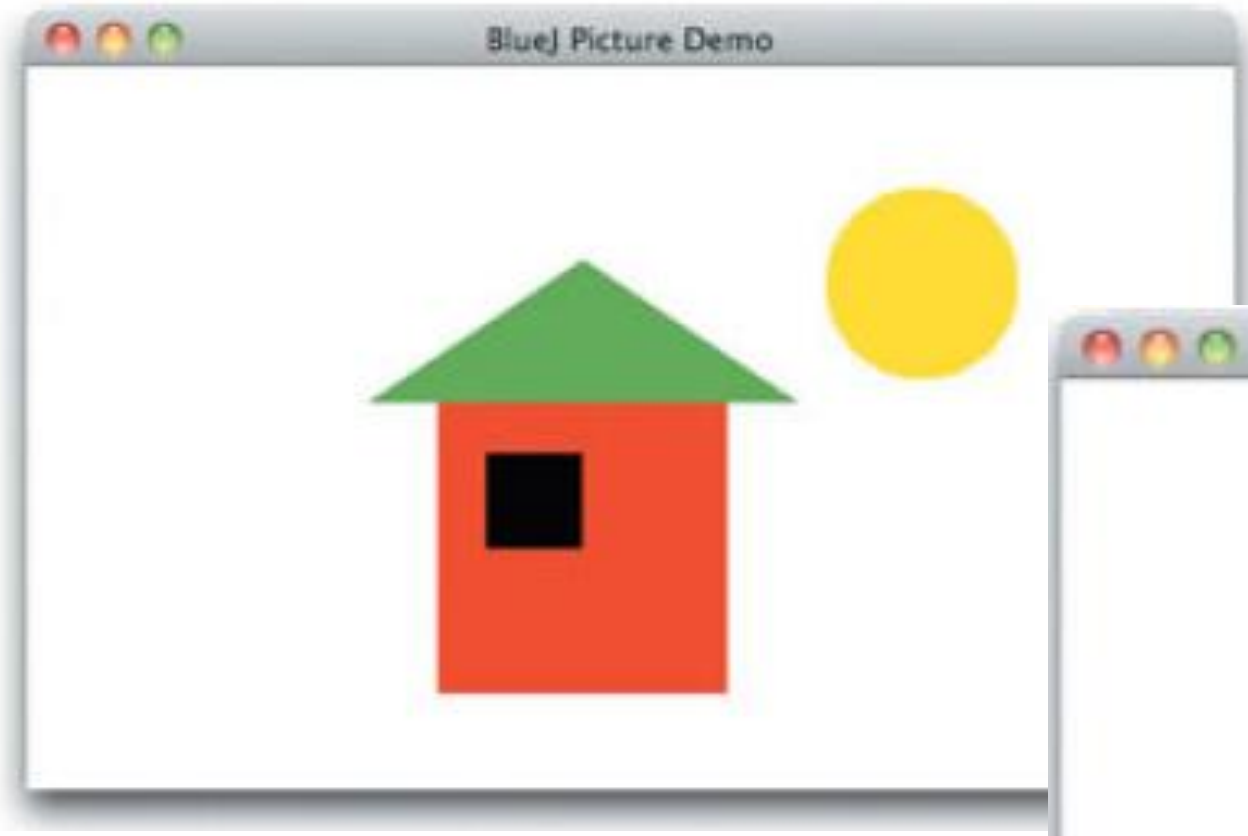
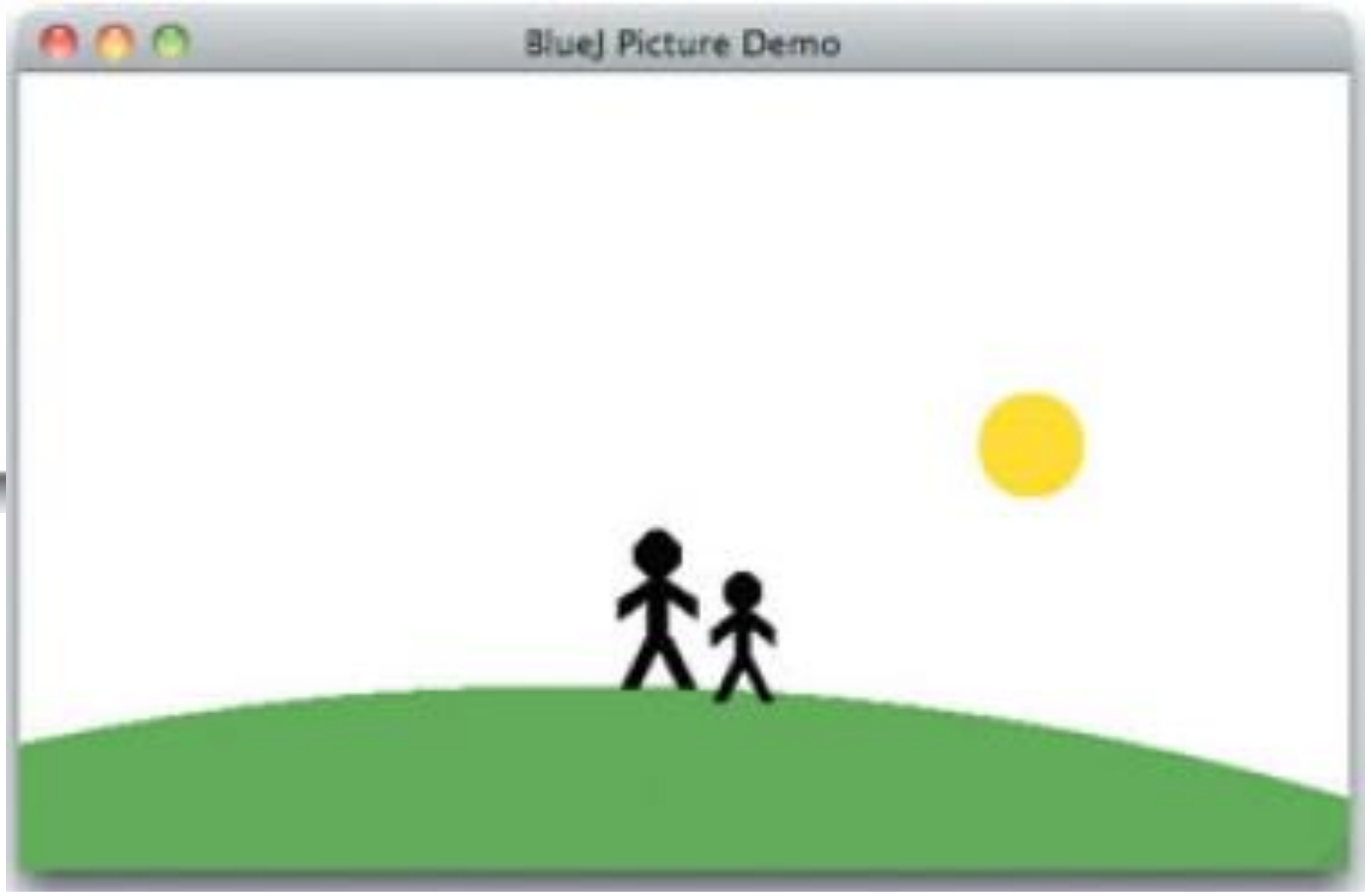
| | |
|---|---|
| private int height | 60 |
| private int width | 70 |
| private int xPosition | 210 |
| private int yPosition | 140 |
| private String color | "green" |
| private boolean isVisible | true |

circle2 : Circle

| | |
|---|---|
| private int diameter | 68 |
| private int xPosition | 110 |
| private int yPosition | 90 |
| private String color | "blue" |
| private boolean isVisible | true |

## What is an object?

- Objects of the same class have the same fields and methods
- Similarly, objects of a different class may have different fields and methods
- Example:
  - Circle has has a “diameter” field


BlueJ Picture Demo

BlueJ Picture Demo

## Exercise 1.9

- Use paper to keep track on how you got it done. You’ll need those for a future exercise. What is the minimum number of steps needed?

## Exercise 1.10

- Select *Show Terminal* from the *View* menu. This shows another window that BlueJ uses for text output. Then select *Record method calls* from the terminal's *Options* menu. This function will cause all our method calls (in their textual form) to be written to the terminal. Now create a few objects, call some of their methods, and observe the output in the terminal window.

## What did we learn from Exercise 1.10?

- We can see what creating an object and giving it a name looks like.
- We see that, to call a method on an object, we write the name of the object, followed by a dot, followed by the name of the method. The command ends with a parameter list—an empty pair of parentheses if there are no parameters.
- All Java statements end with a semicolon.

## Exercise 1.11

- Select *Show Code Pad* from the *View* menu. This should display a new pane next to the object bench in your main BlueJ window. This pane is the *Code Pad*. You can type Java code here.

## Exercise 1.12

- In the Code Pad, type the code shown above to create a person object and call its **makeVisible** and **moveRight** methods. Then go on to create some other objects and call their methods.
- **Tip** You can recall previously used commands in the Code Pad by using the up arrow.

## Exercise 1.13

- Open the *house* project. Create an instance of class **Picture** and invoke its **draw** method. Also, try out the **setBlackAndWhite** and **setColor** methods.

## Exercise 1.14

- How do you think the **Picture** class draws the picture?

## Concept:

- **Method calling**. Objects can communicate by **calling** each other's **methods**.

- So… how do we write the class for such an object?

## Concept:

- The **source code** of a class determines the structure and behavior (the fields and methods) of each of the objects of that class.
  - The text that defines the details of the class
- A large part of learning the art of programming is learning how to write these class definitions

## Exercise 1.15

- Look at the pop-up menu of class **Picture** again. You will see an option labeled *Open Editor.* Select it. This will open a text editor displaying the source code of the class.

## About compilation…

- The computer doesn’t understand Java directly
- The computer uses machine language (also called assembler)
- Compiling translates the Java code into machine code

- Java actually creates virtual machines…

## You are now ready…

- Exercises 1.16-1.20

## Time for the *lab-classes* project

- I’ll start the exploration, then
- You play while doing Exercise 1.21

## Methods that return values

- Methods may return information about an object via a r**eturn value.**
- Like with parameters (values passed into methods), return values can vary based on type.

## Exercise 1.22

- Create some student objects. Call the **getName** method on each object. Explain what is happening.

## Objects as parameters

- Not only can you pass primitive types (int, boolean) into a method, but you can also pass objects (instances of type String, Circle, Student, etc.)

## Exercises 1.23-1.26

- Do it now!
- Big idea: in order to enroll a student, you need to create several instances from the Student class.
- Remember that the student’s name is not the same as the name of the object.

## What are we noticing?

- Private vs. Public
- Constructors
  - The constructor signature vs. method signatures
- Fields
- Debugging tricks
- Curly brackets
- Return values vs. parameters

## Summary

- **object** Java objects model objects from a problem domain.
- **class** Objects are created from classes. The class describes the kind of object; the objects represent individual instantiations of the class.
- **method** We can communicate with objects by invoking methods on them. Objects usually do something if we invoke a method.

## Summary

- **parameter** Methods can have parameters to provide additional information for a task.
- **signature** The header of a method is called its signature. It provides information needed to invoke that method.
- **type** Parameters have types. The type defines what kinds of values a parameter can take.

## Summary

- **multiple instances** Many similar objects can be created from a single class.
- **state** Objects have state. The state is represented by storing values in fields.
- **method calling** Objects can communicate by calling each other’s methods.

## Summary

- **source code** The source code of a class determines the structure and behavior (the fields and methods) of each of the objects of that class.
- **result** Methods may return information about an object via a return value.

## Last Exercises

- 1.30-1.36!

## Review

- Method Signatures
  - Starts with the word “public” or “private”
  - Then the type returned by the method or “void” if the method does not return a value
  - Then the name of the method
  - Then open parenthesis

- Then the inputs to the method
  - If there is no inputs, then finish the method with a close parenthesis
  - Each input is specified by a pair of words
    - A type
    - A variable
      - Variables must start with a letter
      - Variables can not be the same name as the class, method, or any fields for the object
    - A comma separates the type/variable pair, if there is more than one