AP Java

Objects First With Java – A Practical Introduction Using Blue J (2nd edtion)
David Barnes, Michael Kolling

Chapter 1.1

Objects =

- parts used to build a model of some part of the world.
- Specific instance. (I,e. my Toyota).
- Start names of objects with lower case letters (mytoyota)

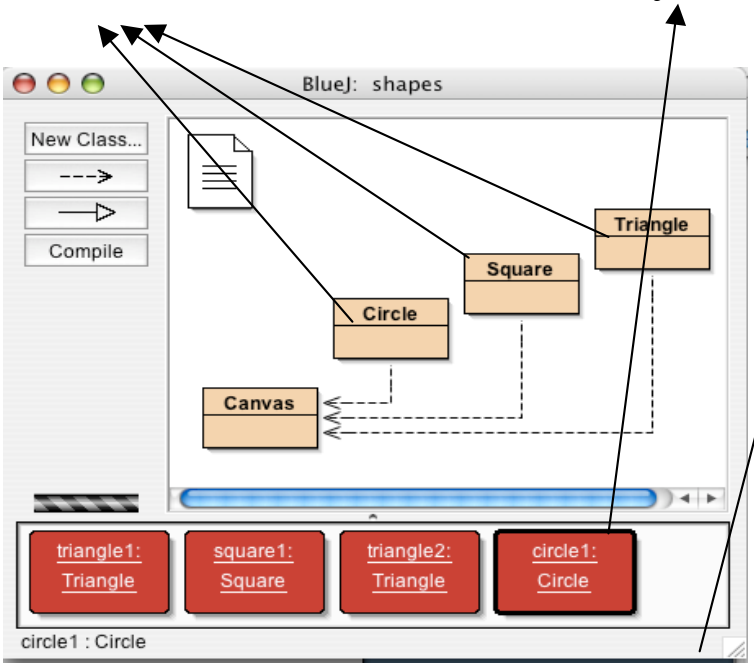
Classes =

- used to categorize objects. General category (i.e. cars)
- Start names with upper case letters (Toyota)

Chapter 1.2

Running BlueJ:

Classes Objects Object Bench



---.> Cntrl/click on object rectangle to create new object.

Chapter 1.3 Calling Methods

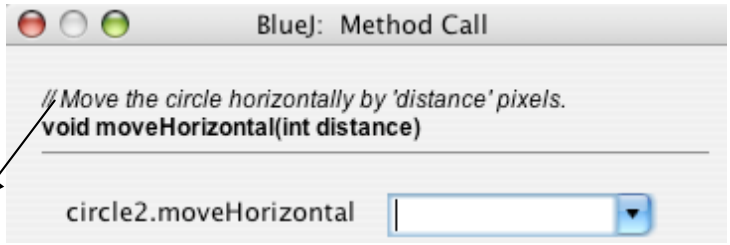
Cntrl/click on object.
Choose operation from drop down menu.
Method = operations that manipulate an object.

Chapter 1.4 Parameters

Paramater = additional values required by some methods.
Signature = the header of a method . Provides information needed to invoke the method.
i.e. void moveHorizontal (int distance)
(type name).

Chapter 1.5 Data Types

type specifices what kind of data can be passed to a parameter.
Int = integer
String = text (always surrounded by quotes).
Comment = in method top of call dialog box. Gives information about to reader.

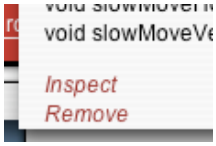


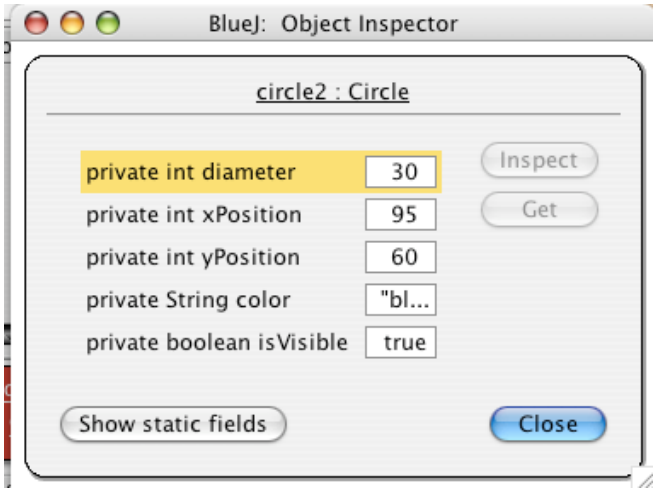
Chapter 1.6 Multiple Instances

- Classes can produce multiple objects (instances) of that class.
- Methods can have more than one parameter
  - i.e. void changeSize (int newHeight, int newWidth) (for triangle)

Chapter 1.7 State

State: The set of values of all attributes defining an object (i.e. x-position, y-postion, color, diameter…) . Represented by storing values in fields. See state with *Inspect* function. See *Object Inspector*

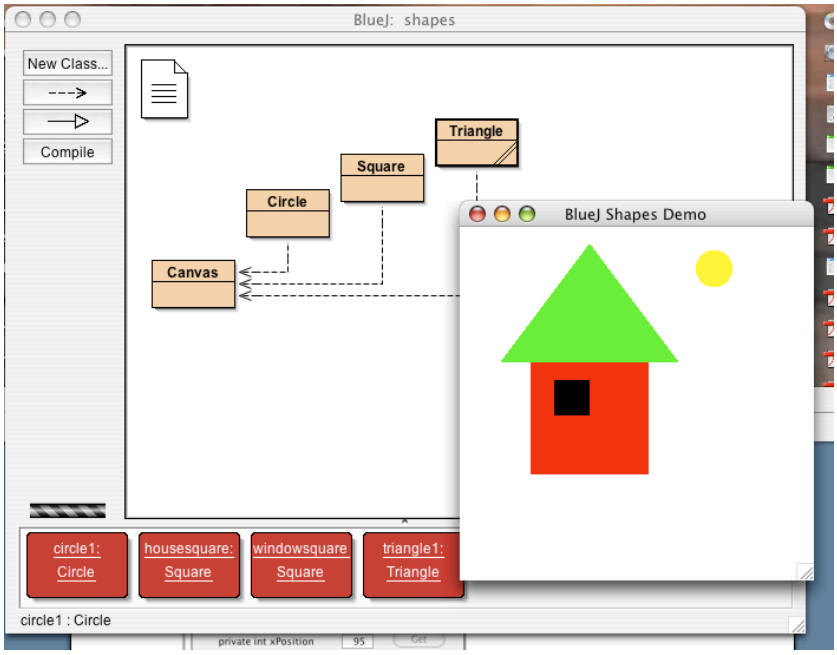




Chapter 1.8 Object Defined

- Objects of same class have same number, type, and names of fields
    - Values of individual field may be different.
- Objects of different classes have different fields (i.e. circles have diameter; triangles have height and width.)
- Number, types and names of fields are defined in the Class. (not in object)
- Anytime an object of a certain class is created, object will have the fields defined in the class.
- Methods are defined in class of an object. Therefore, all objects of same class have same methods.

Exercise 1.9



Chapter 1.9 Object Interaction

- To perform sequence of tasks, create a class to perform sequence (do not do sequence by hand).
- Objects can create other objects and can call each other's methods.

Chapter 1.10 Source Code

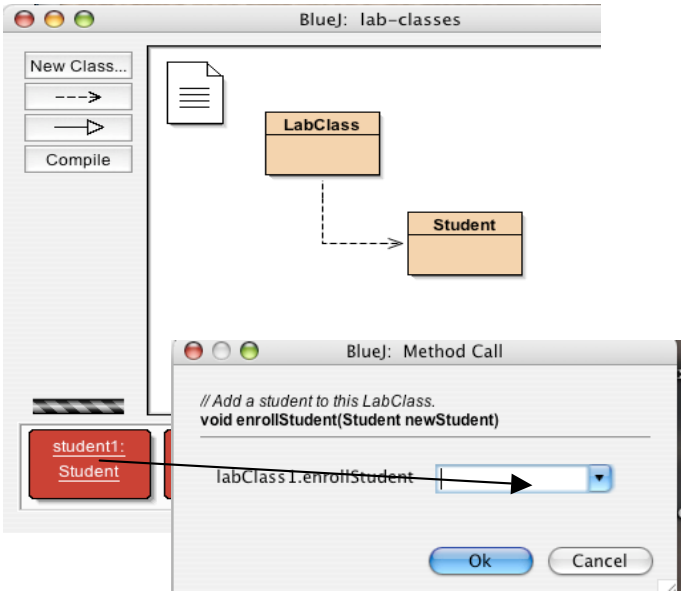
- Each class has specific source code.
- *Source code* = text that defines details of the class. (select *Open Editor* from cntrl/click on Class icon or double click on Class icon)
- Compilation: translates higher order language (java) to binary code (machine language).

Chapter 1.12 Return Values (program Lab Classes)

- Running Methods can return a Results value
- *Results:* Methods may provide information about an object via a *Return Value*.
- *Return Values:* Allow us to get information about an object via a method call (i.e. in Lab Classes program, calling method *string getNam ()* returns name typed into parameter when object was created in Class: Student.
- Methods can (1) change state of object or (2) found out about a state.

Chapter 1.13 Objects as Parameters (program Lab Classes)

- Objects (i.e. *student1*) can be passed as parameters to methods of other objects
- When methods expects another object as parameter, expected object's class name is specified as parameter type in method signature)
- To do:
  - Create new object (labClass1) from LabClass object
  - Call method void *enrollStudent(Student newStudent),*
  - In Method Call window, Click on *student1:* object. Student info will be placed into parameter *enrollStudent(Student newStudent),*



Chapter 1.14 Summary

- Classes provide definitions for objects
- Classes = general concept of a thing.
- objects represent concrete instance of a Class. (i.e. Mammal = Class; lion = object)
- Objects use *methods* for communication
  - Make changes in state or Gain information about object.
  - Parameters = information to run method. Parameters have types (i.e. string, int, ….) ((type: void = expect no return)).
  - Fields = where objects store data.
  - State = all data found in an objects's fields.

Create method:

public int average(int firstNumber, int secondNumber)

Type of return expected

Name of method

signature

Vocab: object, instance, method, signature, parameter, type, state, source code, return value, compiler

Ch 1 appendix:

Source Code:

Circle Class definition for a **Circle**

Attributes

```
private int diameter;
private int xPosition;
private int yPosition;
private String color;
private boolean isVisible;
```

diameter - determines the size of the circle
xPosition & yPosition - determines the placement of the circle
color - determines the colour of the circle
isVisible - determines whether the circle is visible or not

Constructor

```
public Circle()
{
diameter = 30;
xPosition = 20;
yPosition = 60;
color = "blue";
isVisible = false;
}
```

The constructor method for **Circle** takes no parameters. It assigns the following values:
diameter 30
xPosition 20
yPosition 60
color "blue"
isVisible false
Note that invoking the **Circle** constructor method **twice** will result in the creation of **2 separate Circle objects** but the **state** of these two objects will be **exactly the same** - they will both have a **diameter** of **30**, an **xPosition** of **20**, **yPosition** of **60**, **colour blue** and be **invisible**.

Answers to Exercises:

Exercise 1.9

The main building:

* Create a new Square object
* Invoke its method makeVisible()
* Make the square bigger by invoking the method changeSize(newSize) (100 is a good size)
* Move the square down by invoking the method moveVertical(distance) (again 80 is a good value)

The window:

* Create a new Square object.
* Invoke its method makeVisible()
* Change its color by invoking changeColor()
* write "black" in the popupwindow
* Move the square down by invoking the method moveVertical(distance) (100 is a good value)
* Move it to the right by invoking moveRight()

The roof:

* Create a new triangle object.
* Invoke its method makeVisible()
* Change its size with changeSize(newHeight, newWidth) (50,140)
* moveVertical(70)
* moveHoizontal(60)

The Sun:

* Create new Circle object.
* Invoke its method makeVisible()
* Change its color by invoking changeColor() (write "yellow" in the popupwindow) Optionally change its size with changeSize(60)
* Move it to the right by invoking moveHorizontal(180)

Exercise 1.11

* It uses the objects of the classes Circle, Square and Triangle.
* It then moves these objects to the right places and changes the sizes and colors of the obejcts. Essentially calling the same methods as used in exercise 1.9

Exercise 1.15

After the line sun.makeVisible() insert the following:

* sun.slowMoveVertical(250);
* Compile the Picture class (Press compile in the editor window)
* Create instance of class Picture and invoke its draw() method.

Exercise 1.16

Remove the line (if added in the previous exercise): slowMoveVertical(250); Right below the last } after the draw() method, add the sunset() method :

```
/**
 * Animates the sunset.
*/
public void sunset() {
   sun.slowMoveVertical(250);
}
```

Compile! And run it.

Exercise 1.18

When calling the method getName(), the name of the student is displayed in a popup window. The name displayed is the one typed in when the object was created.

Exercise 1.20

It shows the number of students in the labclass which is zero.

Exercise 1.27

0	int
"hello"	String
101	int
-1	int
true	boolean
"33"	String
3.1415	double

Exercise 1.28

First you would have to decide which type the field should have. String would be a good type to hold a name, so we add the following line to the source file of Circle:

```
   private String name;
```

The above line could be placed after this line in Circle.java:

```
   private boolean isVisible;
```

Exercise 1.29

```
public void send(String msg)
```

Exercise 1.30

public int average(int firstNumber, int secondNumber)

Exercise 1.31

The book is an object. The class could be book.

Exercise 1.32

Yes, an object can belong to several classes. One of the more famous examples are the platypus, which is both a mammal and egg-laying.