CITS1001 week 1
Objects and Classes

Arran Stewart

March 1, 2018

Fundamental concepts

- Class
- Object
- Instance

- method
- parameter
- signature

Reading: Chapter 1 of Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Objects and classes

- Class
  - A class represents a general category of things
  - e.g. Car, Bicycle, Student, Dog
- Object
  - Individual objects are created from a class
  - Objects represent ‘things’ from some real-world problem domain
  - e.g: “the red car down there in the car park”
- Instance
  - Any particular object will be an *instance of* some class

Example - student

- The set of all students forms the *class* Student
- A class describes features held in common
- Each individual student is an *object* of the class Student
    - e.g. John Smith and Janice Lee are instances of Student

Example - Dog

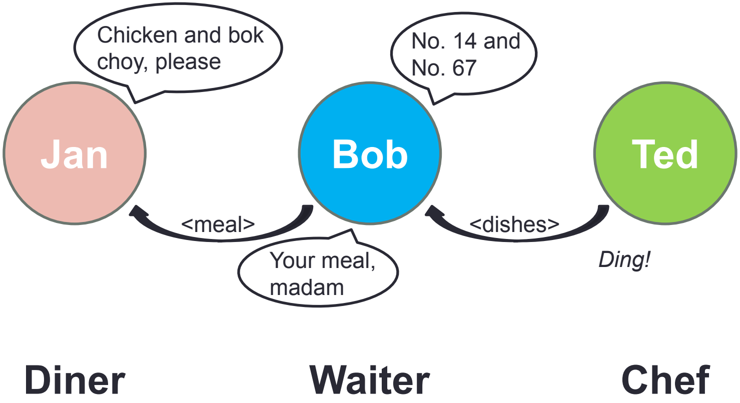
- The set of all dogs forms the class Dog
- Each individual dog is an object of the class Dog
- Spot, Rover, and Rex are all instances of Dog

Example – Restaurant

- Writing programs is largely about managing *complexity*
- How is something complex organized in the real world?
- Consider a restaurant . . .
  - Diners want meals
  - Chefs prepare dishes
  - Waiters take orders, and bring food to the tables
  - Busboys collect and wash plates
  - Barmen prepare and serve drinks
  - The maitre'd makes reservations and seats diners

Each type of person provides a narrow range of services. The restaurant involves the co-operative interaction of all the restaurant staff and clients.

A Restaurant



Objects - what is a Waiter?

In this scenario, a Waiter has the following *actions* that can be performed:

- Bring menus
- Take orders
- Bring meals

We can deal with any individual waiter, whether we have met them before or not, based solely on our knowledge of what things a Waiter can do.

Review (1)

- Class
  - Objects are created from classes. The class describes the kind of object; the objects represent individual instantiations of the class.
- Object
  - Java objects model specific objects from the problem domain.
- Instance
  - Many similar objects can be created from a single class.

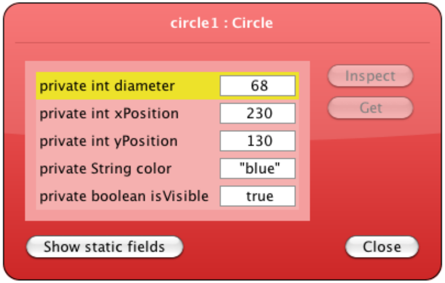
Review questions

- Can a class have several different objects? *Discuss.*
- Can an object have several different classes? *Discuss.*

What's in an object?

- Objects have *operations* that can be invoked
  - Java calls these **methods**
  - An object usually *does* something when we invoke a method (or gives us some information)
- Objects have **state**
  - The state is represented by the stored values of attributes in "fields"
  - The state of an object is a "snapshot" of that object at a particular moment in time
- e.g. the class Student might have
  - An attribute studentNumber, that never changes, and
  - An attribute booksBorrowed, that does change

State for a Circle object


circle1 : Circle

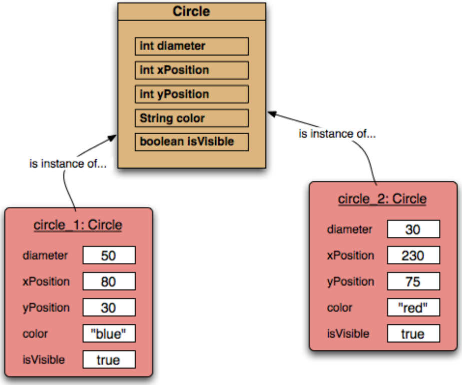
private int diameter	68
private int xPosition	230
private int yPosition	130
private String color	"blue"
private boolean isVisible	true

- Notice the **types** of the fields this circle object has:
  int, String, boolean
- Types restrict the values that a field can take
  We might want to specify that 50 is a valid value for the diameter of a circle, but “blue” is not
  We will look more at types in future weeks.

Instances

Many instances can be created from a single class.

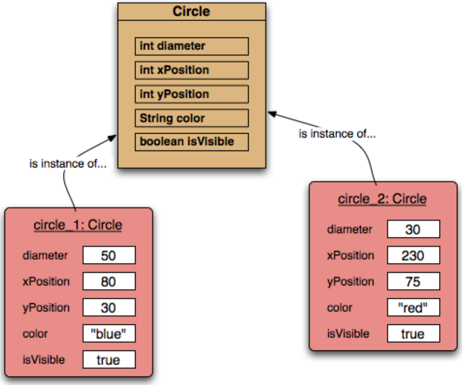
The class defines what fields an object has, but each object stores its own set of values (the state of the object).



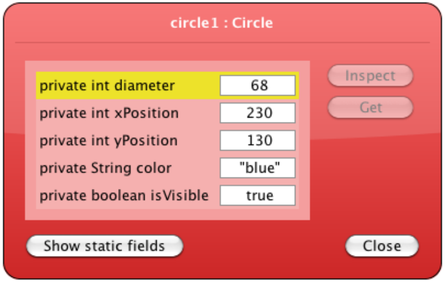
Instances and objects

In Java, we say that each object is an *instance of* some class.

In this case, both the objects circle_1 and circle_2 are instances of the Circle class



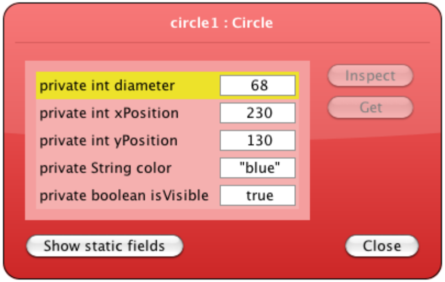
Methods and parameters for a Circle object


circle1 : Circle

private int diameter	68
private int xPosition	230
private int yPosition	130
private String color	"blue"
private boolean isVisible	true

- Methods correspond to things we might “ask” an object to do

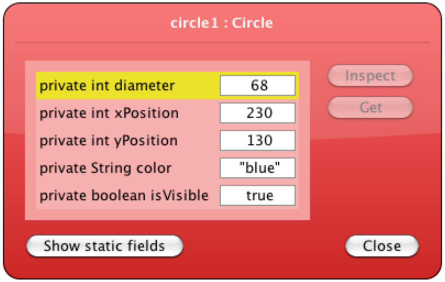
Methods and parameters for a Circle object


circle1 : Circle

private int diameter	68
private int xPosition	230
private int yPosition	130
private String color	"blue"
private boolean isVisible	true

- Methods correspond to things we might "ask" an object to do
  - Given the above attributes for a Circle object, what methods might it have?

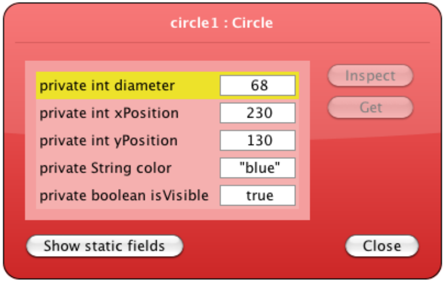
Methods and parameters for a Circle object



private int diameter	68
private int xPosition	230
private int yPosition	130
private String color	"blue"
private boolean isVisible	true

- Methods correspond to things we might "ask" an object to do
    - Given the above attributes for a Circle object, what methods might it have?
- Methods may have **parameters** which pass additional information needed to perform a task.

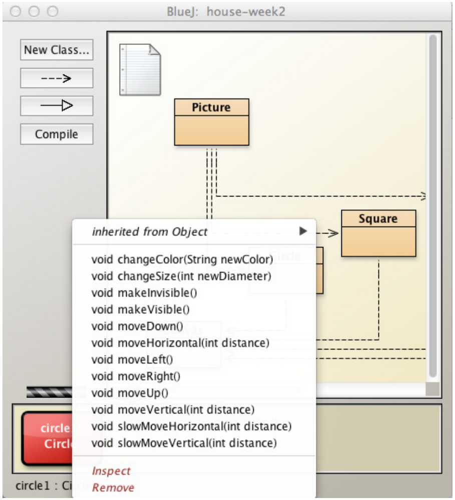
Methods and parameters for a Circle object


circle1 : Circle

private int diameter	68
private int xPosition	230
private int yPosition	130
private String color	"blue"
private boolean isVisible	true

- Methods correspond to things we might “ask” an object to do
  - Given the above attributes for a Circle object, what methods might it have?
- Methods may have **parameters** which pass additional information needed to perform a task.
  - Given the methods we have identified, what parameters would they have?

Methods and parameters of a Circle object (2)



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)

Method signatures

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

- The name of a method, together with the types of the parameters, are called the *signature* of the method.
  The method signature provides information needed to invoke that method.

Method signatures

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

- The name of a method, together with the types of the parameters, are called the *signature* of the method.
  The method signature provides information needed to invoke that method.
- Q: What is the signature of the changeSize method?

Method signatures

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

- The name of a method, together with the types of the parameters, are called the *signature* of the method.
  The method signature provides information needed to invoke that method.
- Q: What is the signature of the changeSize method?
- Q: What is the signature of the moveDown method?

Method signatures

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

- The name of a method, together with the types of the parameters, are called the *signature* of the method.
  The method signature provides information needed to invoke that method.
- Q: What is the signature of the changeSize method?
- Q: What is the signature of the moveDown method?
- Q: What are the differences between the signatures of the slowMoveHorizontal and slowMoveVertical methods?

More on methods

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

- Parameters pass additional information needed to execute a method
  - i.e., they act as “input” to the method
- Parameters have types.
  - The type defines what kinds of values a parameter can take.
- Methods may also return a *result* via a return value.
  All the methods above have the “void” return type, indicating they “do” things, rather than returning information.
  But we will see non-void return types in future lectures.
- Objects **communicate** by calling each other’s methods

Source code

- Each class has source code (Java code) associated with it that defines its details (fields and methods).
- We will start looking inside source code next week

Source code (2)

- The source code of a class specifies three things

- What is in the state of each object
  - i.e. what fields it has
- The *behaviour* of each object
  - i.e. what methods it has, and what they do
- How objects are created
  - The state is initialized by a *constructor*

- A Java application is a collection of classes

Review (2)

- Method
  - We can communicate with objects by invoking methods on them. Objects usually do something if we invoke a method.
- Parameter
  - Methods can have parameters to provide additional information for a task.
- Type
  - Parameters have types. The type defines what kinds of values a parameter can take.
- Signature
  - The name of a method, together with the types of its parameters, is called its signature. It provides information needed to invoke that method.

Review (3)

- State
  - Objects have state. The state is represented by storing values in fields.
- Method calling
  - Objects can communicate by calling each other's methods.
- Source code
  - The source code of a class determines the structure and behaviour (the fields and methods) of each of the objects of that class.
- Result
  - Methods may return information about an object via a return value.

Review questions

- Write the signature for a method named addStudent that has one parameter of type String and does not return a value.

Challenge question:

- Write the signature for a method named max that has two parameters both of type int, and returns an int value.

Summary of concepts introduced

- You should now be able to give an explanation of each of these terms:
  - object;
  - class;
  - method;
  - parameter;
  - signature;
  - type;
  - state;
  - source code

Review these concepts

- You can watch watch David Barnes' Intro to Key Concepts here:

  https://www.youtube.com/watch?v=CPUaTT0Xoo4&list=PLYPWr4ErjcnzWB95MVvlKArO6PIfv1fHd&index=1