Objects and Classes
Lecture 2

Waterford Institute of Technology

January 27, 2015

John Fitzgerald

Classes and Objects

Example of class

- Abstract description or specification of some entity
- Example
  - Car with following specification:
    - Make
    - Model
    - Color

Example of object

- Specific instance of an entity defined by its class
- Example: car object
  - Make: VW
  - Model: Golf
  - Color: red

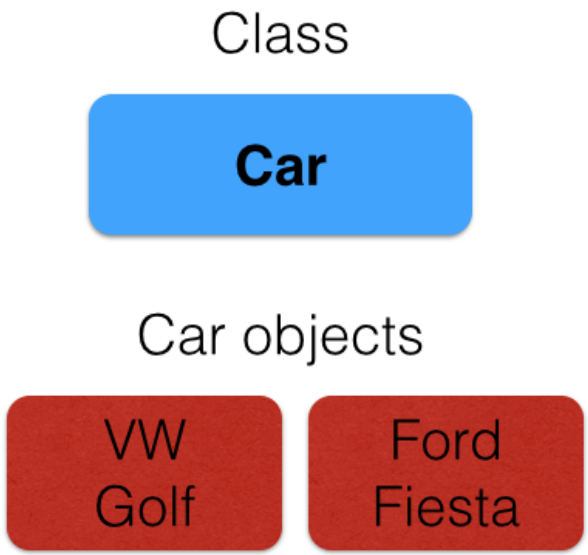
Concepts

Class

- Describes, defines or specifies objects
- Car class broadly descriptive
  - Actual make, model not known to class

Car object clearly specified

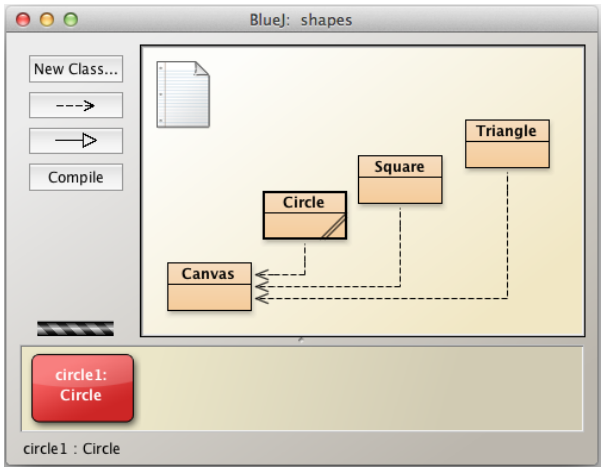
- Object created in conformance with class specification
- Exact make, model known



Create objects with BlueJ

Open Shapes project in BlueJ

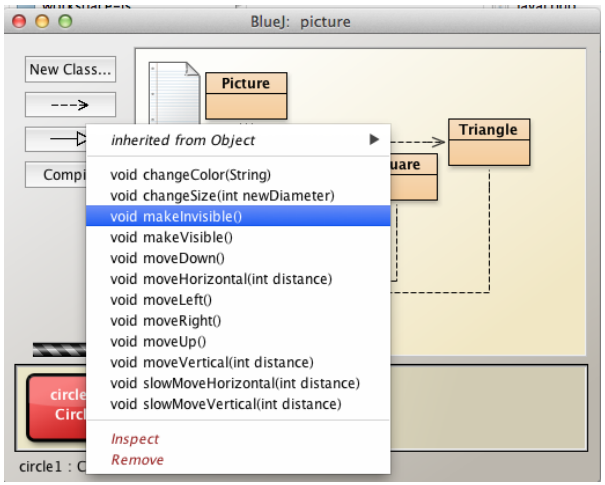
- Select Circle class & right click
- Choose new Circle
- Accept default name
- Circle object now on object workbench



Method invocation

Still working with Shapes project

- Right click on Circle object
- Select makeInvisible
- Object disappears
- Select makeVisible
- Object reappears



Method description

What is a method?

- A program within the class
- Methods
  - Perform actions
  - Can optionally return data
- Circle method actions:
  - Make circle object appear
  - Make circle object disappear
  - Change color circle object
  - Change size circle object

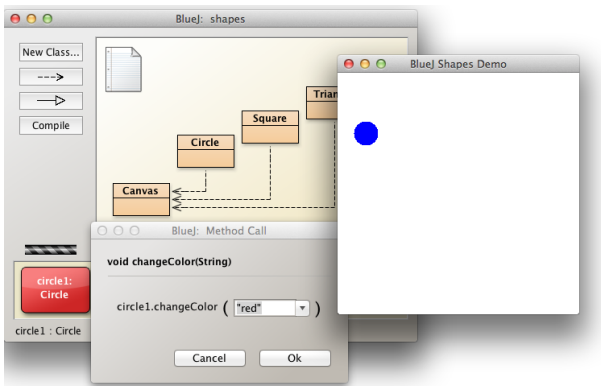
void: no data returned

```
void makeVisible();
void makeInvisible();
void changeColor(String);
void changeSize(int newDiameter);
```

Parameters

What are parameters?

- Some methods require no further information.
  - Example: *makeInvisible()*
- Others require additional information.
  - Additional information termed parameters
  - Zero, one or more parameters permitted.
  - Example *changeColor(String)*
  - String a Java object representing new color.



Method Signature

Provides information needed to call method

Method signature comprises method name & parameter list

- Method: *public void changeColor(String a)*.
- Signature:*changeColor(String a)*.
- Excludes return type (void)
- Excludes access modifier (public)

```
public void changeColor(String color)
{
  this.color = color;
}
```

Method Signature

Another example

- *public int getCredits()*.
- Signature: *getCredits()*.

```
int getCredits()
```

Method Signature

Class methods must have different signatures

Illegal: method signatures the same

```
public class Circle {
  public void setColor(String c) {this.color = c;}
  public String setColor(String c){this.color = c; return this.color;}
}
```

Legal: method signatures different

```
public class Circle {
  public void setColor() {this.color = "red";}
  public void setColor(String c){this.color = c;}
}
```

Data types

Signature of method:

- Informs number of parameters
- Informs type of each parameter
- Eight primitive data types, e.g:
  - int: represents integer values, e.g. 10, 25
  - boolean: may be *true* or *false*
- String: is an object. Represents text, e.g. "color", "10"

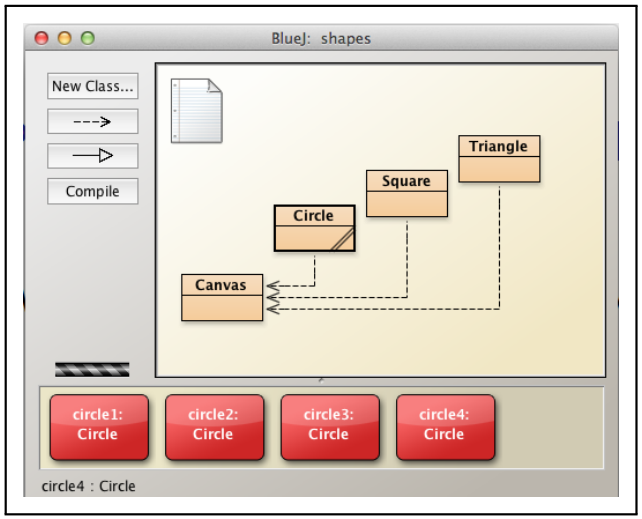
Some primitive Java types

int
float
double
long
boolean

Multiple instances

Many objects may be created from single class.

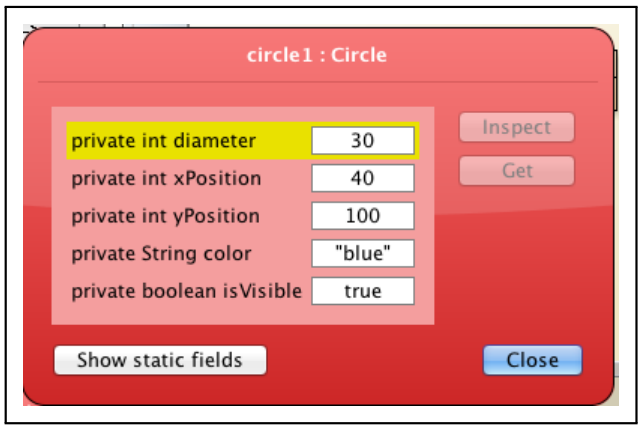
- Object: an instance of a class
- Instantiating class produces object
- Each object can have own set of internal data



State

Objects have state.

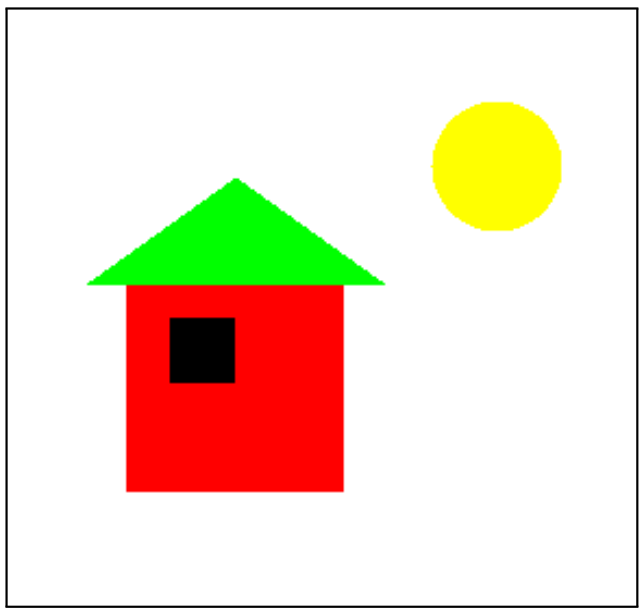
- Class Circle has fields
- Circle object field *color* has value (attribute)
  - Example: "blue"
- Object state: set of all values of all fields



Object interaction

Could create Picture manually
Or could be created by program

- Picture Class instance creates
  - Two Square objects
  - One Triangle object
  - One Circle object
- Objects' states determine
  - Size of each object
  - Position of each object
  - Color or each object



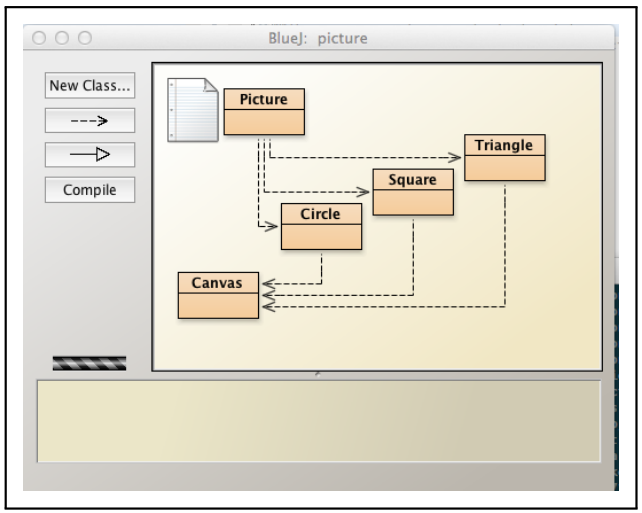
How Picture object created

Picture class contains

- Square class (wall)
- Square class (window)
- Triangle class (roof)
- Circle class (sun)

Picture has a method *draw* that

- Instantiates these classes
- Sets the state of each object



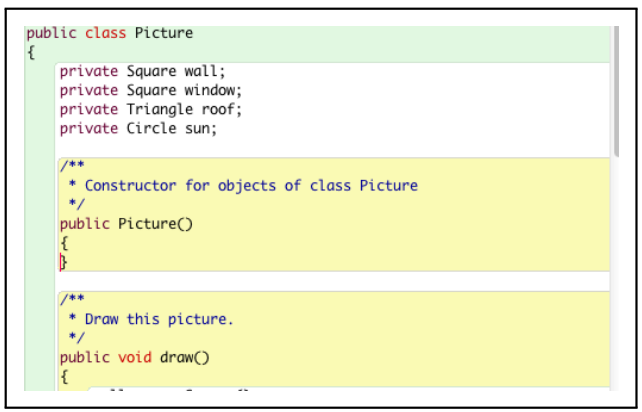
Picture object source code

Source code Java text
Defines fields and methods

- private `Square wall`;
- public void `draw`();

When source code compiled

- Object can be created
- State can be changed
- Object methods callable



```
public class Picture
{
    private Square wall;
    private Square window;
    private Triangle roof;
    private Circle sun;

    /**
     * Constructor for objects of class Picture
     */
    public Picture()
    {
    }

    /**
     * Draw this picture.
     */
    public void draw()
    {
```

Compilation

Source code is compiled

Computer processor requires binary (machine code)

- 00110001110101011

Difficult programming in binary

Hence human readable Java

Compiler: source code to machine code

Changed source requires recompilation

Bits and Bytes

Bit (**B**inary **D**igit): smallest unit of compilation

- Value range 0, 1

Byte: 8 bits

- 00000000
- 01001101

MegaByte (MB):

- 1024 bytes

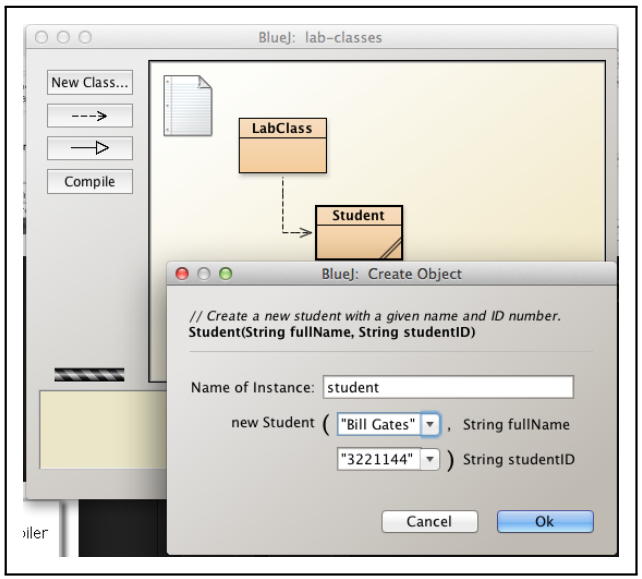
Megabit (Mb):

- 1024 bits
- 1 megabyte (MB)

Using parameters when creating objects

Student project example

- Create new student
- Object name required
- Parameters required
  - *String fullName*
  - *String studentID*



Object state

Student object state

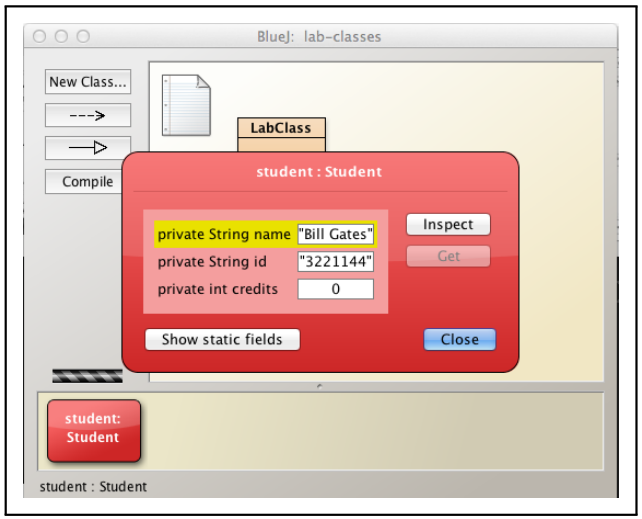
- private String name : "Bill Gates"
- private String id : "3221144"
- private int credits : 0

Notice double quotes

- These required for String objects

Notice third field undefined

- Assigning value here later task



Return values

Notice Student class methods
Some methods return data

- *String getName()*

Method *getName* when invoked

- Sends back String object
- String object contains student name

Signature of method informs return type

- *void* means no value returned
- *int* means an integer returned

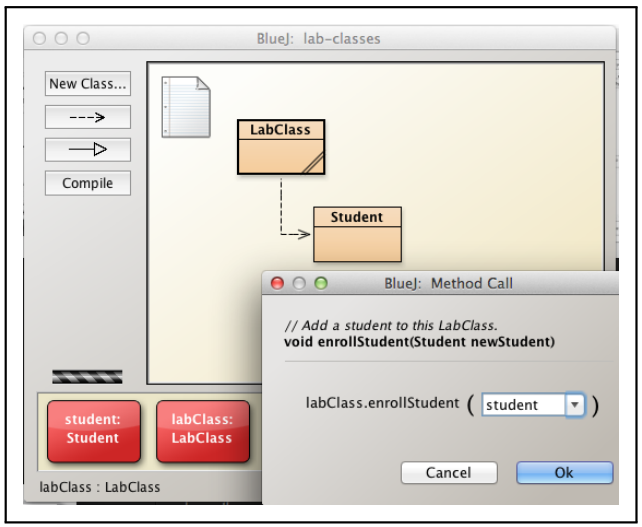
```
void addCredits(Int additionalPoints);
void changeName(String replacementName);
int getCredits();
String getLoginName();
String getName();
String getStudentID();
void print();
```

Objects as parameters

Parameters may be

- Primitive data types (example: int, float)
- Objects (example: String)

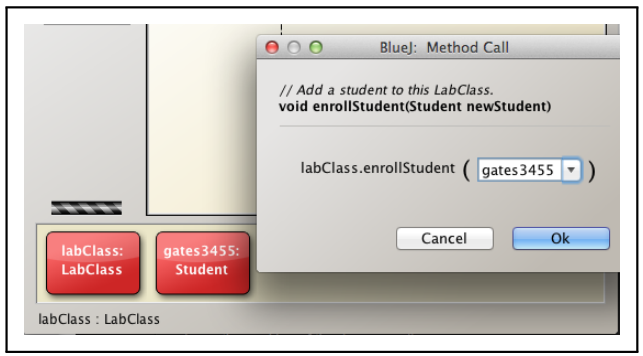
LabClass has students
Enrolling new student passes Student object as parameter



Objects as parameters (continued)

Objects can be passed as parameters

- *Student gates3455*
- *labClass.enrollStudent(gates3455);*
- Notice no double quotes
- labClass is LabClass object
- gates3455 is Student object

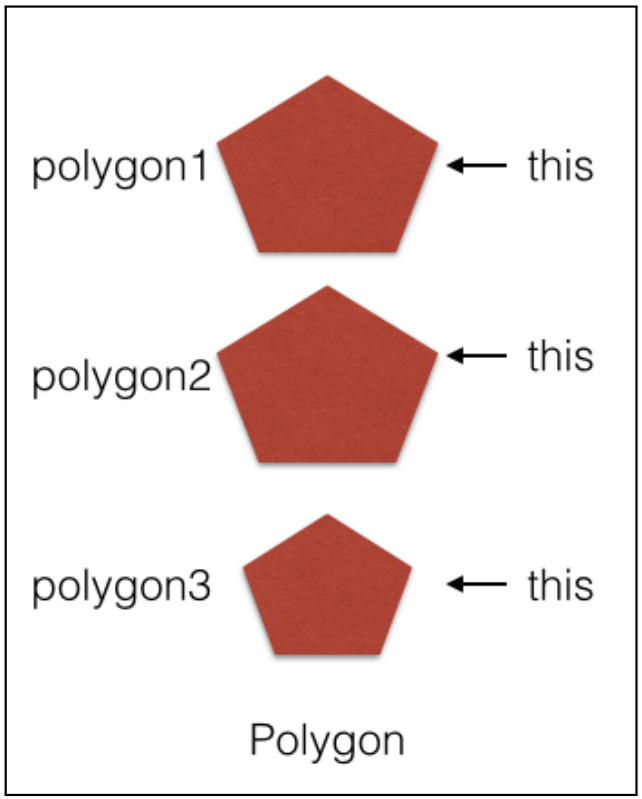


The *this* keyword

Object reference

- Memory address where object stored
- Address accessible by *this* keyword
- Common usage where field shadowed by parameter

```
public class BankAccount {
    int sum;
    public BankAccount(int sum) {
        this.sum = sum;
    }
}
```



The *this* keyword

Here is class Student constructor as written in the BlueJ example

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

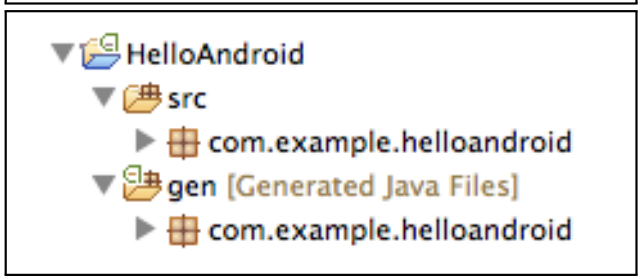
Here is an alternative approach using the *this* reference.

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

Package

Package definition

- A grouping of related types
- Example: a folder of class files
- One benefit to provide access protection



Controlling access

Access modifier

- Determines other class access to field or method

Fields may be declared thus:

- int value;
- public int value;
- private int value;
- protected int value;

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

Table 1 : Access Levels

Block

Block is code between curly braces.

```
public Tree(int val)
{
  this.val = val;
}
```

Blocks can be nested.

Example of method block enclosed by class block:

```
public class Student
{
  String name;
  public String getName()
  {
      return name;
  }
}
```

Scope

Scope refers to lifetime and accessibility of variable

```
public class Tree
{
    int val;
    ...
    public Tree(int val)
    {
        this.val = val;
    }
}
```

this.val

- Has class scope
- Visible (usable) throughout class

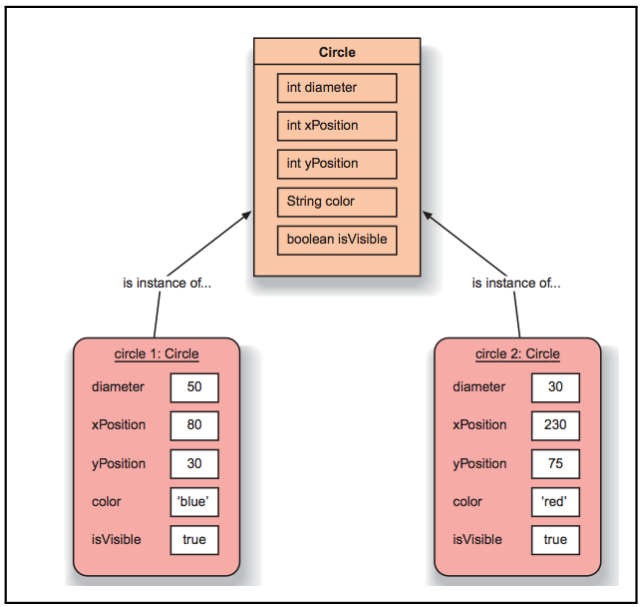
val

- Has local scope
- Visible (usable) only within constructor

Summary

Classes and Objects

- Object is instance of class
- Objects of same class have same fields
- Field types & names declared in class
- Field values set when object created
- Field values typically differ across objects



Summary

Classes and Objects

- Class represents general concept
- Several objects creatable from single class
- Objects store data in fields
- Object state comprises all data values
- Objects have methods
- Methods can change object state
- Methods can retrieve information from objects

Summary

Classes and Objects

- Method : method invocation communicates with objects
- Return value : data sent to caller when method invoked
- Signature : header of method facilitating invocation
- Parameter : data passed to method
- Type : defines kind of data
- State : set of field values (attributes) in object
- Source code : Java language description of program
- Compiler : software program converts source code to bytecode