Week 1
Objects and Classes

suggested reading:

Textbook, Ch. 1

Plan For Today

- Introduction
- Objects and Classes
- What is an object?
- Demo
- Methods and Parameters

Plan For Today

- Introduction
- Objects and Classes
- What is an object?
- Demo
- Methods and Parameters

What is programming?

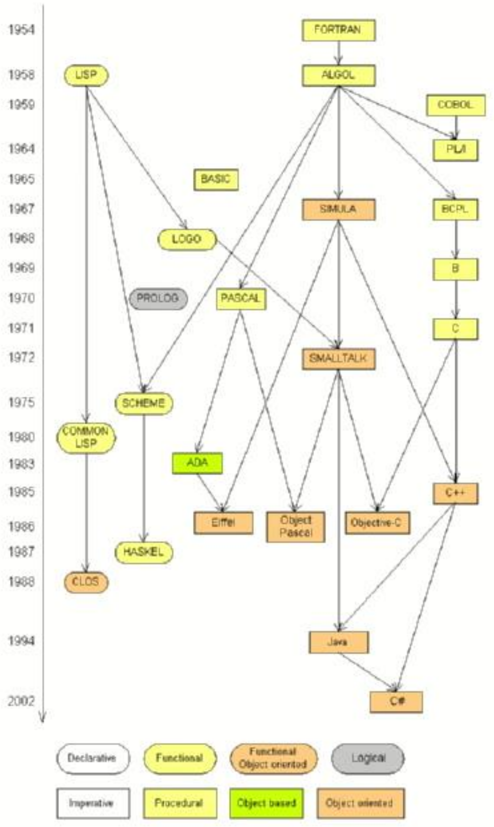
- **program**: A set of instructions to be carried out by a computer.

- **program execution**: The act of carrying out the instructions contained in a program.

- **programming language**: A systematic set of rules used to describe computations in a format that is editable by humans.

Programming languages

- *procedural languages*: programs are a series of commands
  - **Pascal** (1970): designed for education
  - **C** (1972): low-level operating systems and devices
- *functional programming*: functions map inputs to outputs
  - **Lisp** (1958) / **Scheme** (1975), **ML** (1973), **Haskell** (1990)
- *object-oriented languages*: programs use interacting "objects"
  - **Smalltalk** (1980): first major object-oriented language
  - **C++** (1985): "object-oriented" improvements to C
    - successful in industry; used to build OSes such as Windows
  - **Java** (1995): designed for embedded systems, web apps
    - Runs on many platforms (Windows, Mac, Linux, cell phones...)
    - The language taught in this course and our textbook



Object-oriented Programming

- It is a programming paradigm based on the concept of "objects".

- Objects can contain data, in the form of:

  - **Fields** (also known as attributes or properties)

  - **Code**, as methods(also know as procedures).

- Object oriented technology allows the designer to create more robust, reusable software that is easier to test, maintain, refine, and extend.

Plan For Today

- Introduction
- Objects and Classes
- What is an object?
- Demo
- Methods and Parameters

Objects and Classes

- When writing a computer program in an object-oriented language, you are creating, in your computer, a model of some part of the world.
- Java **objects** model objects from a problem domain.
- They may be:
  - Words and paragraphs in a word processor.
  - Users and messages in a social-network.
  - Monsters and heroes in a computer game.
- Objects are created from **classes**. The class describes the **type/category** of object; the objects represent individual instances of the class. “Instance” is roughly synonymous with “object”.

Classes Are Like Blueprints

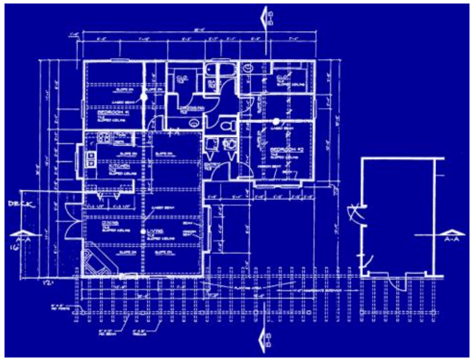
iPod blueprint (class)

state:

song
volume
battery life

behavior:

power on/off
change station/song
change volume
choose random song



constructs

iPod (object) #1

state:
song = “Follow Your Arrow”
volume = 17
battery life = 40%

behavior:
power on/off
change station/song
change volume
choose random song

iPod (object) #2

state:
song = “Tightrope”
volume = 9
battery life = 80%

behavior:
power on/off
change station/song
change volume
choose random song

iPod (object) #3

state:
song = “Burn”
volume = 24
battery life = 25%

behavior:
power on/off
change station/song
change volume
choose random song

Plan For Today

- Introduction
- Objects and Classes
- What is an object?
- Demo
- Methods and Parameters

What is an object?

- Objects have *operations* that can be invoked
  - Java calls them **methods**
  - An object usually does *something* when we invoke a method
- Objects have a **state**
  - The set of values stored in the fields of an object are together called the state of the object
  - You can think of the state of an object as a “snapshot” of that object at a particular moment in time
- e.g. the class Student might have
  - An attribute studentNumber, that never changes
  - An attribute booksBorrows, that does change

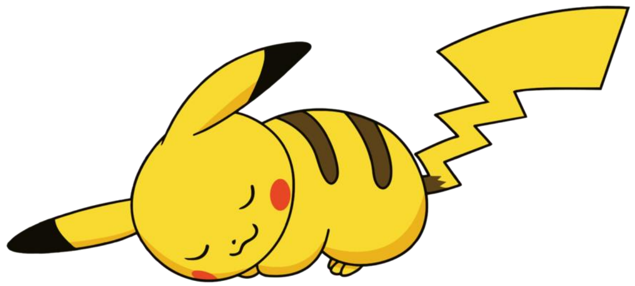
Example - Pokemon

- The set of all pokemons forms the *class* Pokemon
  - A pokemon have attributes such as: name, type, weakness, height, weight, sound.
  - A pokemon have methods such as: attack and talk.
- Each individual pokemon is an object of the class Pokemon
  - Pikachu, Bulbasaur, Charmander are all instances of the class Pokemon

Example

Name: Pikachu
Type: Electric
Weakness: Ground
Height: 0.4m
Weight: 6kg
Sound: Pika, pika

Attack: Performs Thundershock
Talk: Produces a "pika, pika" sound



Objects and Classes

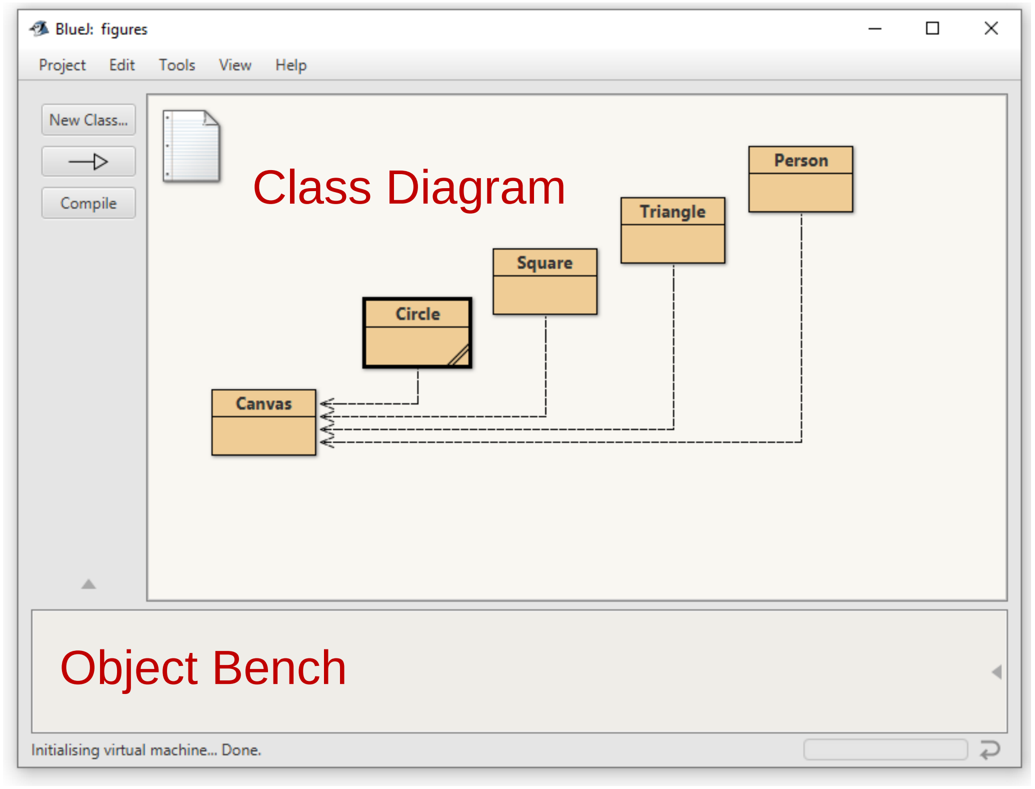
- **Class**
  - A class represents a general kind/category of things
  - E.g. Car, Bicycle, Dog, Pokemon

- **Object**
  - Individual objects are created/instantiated from a class
  - An object represent a particular “thing” from some real-world domain
  - E.g: “The Ferrari down the street is mine”

- **Instance**
  - Any particular object will be an instance of some class

Demo

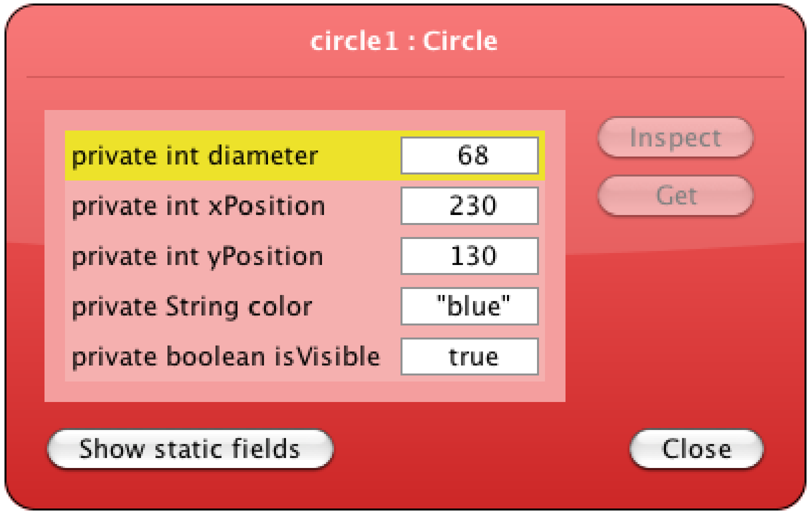
Demo: figures



- Download BlueJ from: www.bluej.org
- Download the necessary code from: https://bit.ly/3dkl6UO

State of a Circle Object

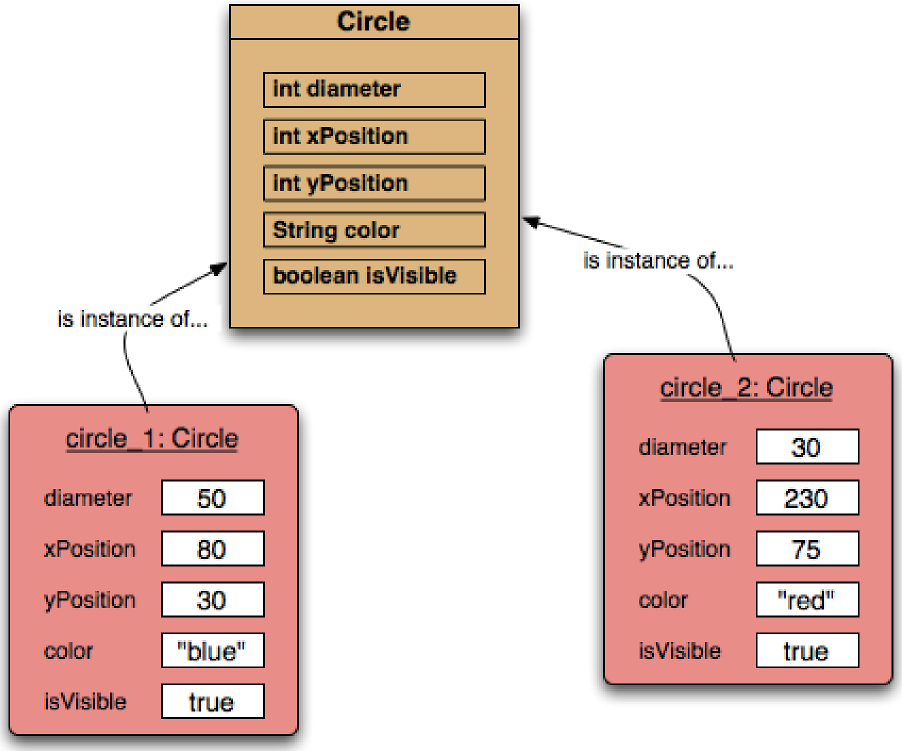
- Notice the **types** of the fields this circle object has:
  - int, String, Boolean
- Types restrict the values that a field cant take
  - We might want to specify that a value such as 50 is a valid value for the diameter of a circle, but “blue” is not.


circle1 : Circle

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

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 (state).

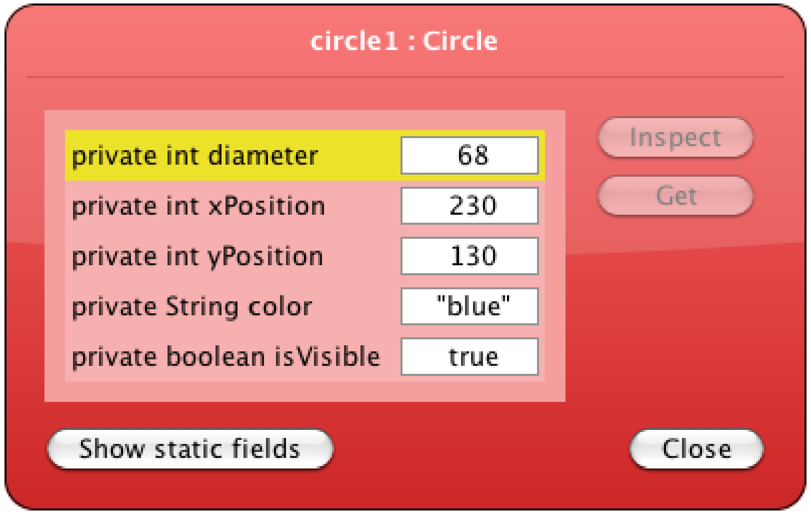


Plan For Today

- Introduction
- Objects and Classes
- What is an object?
- Demo
- Methods and Parameters

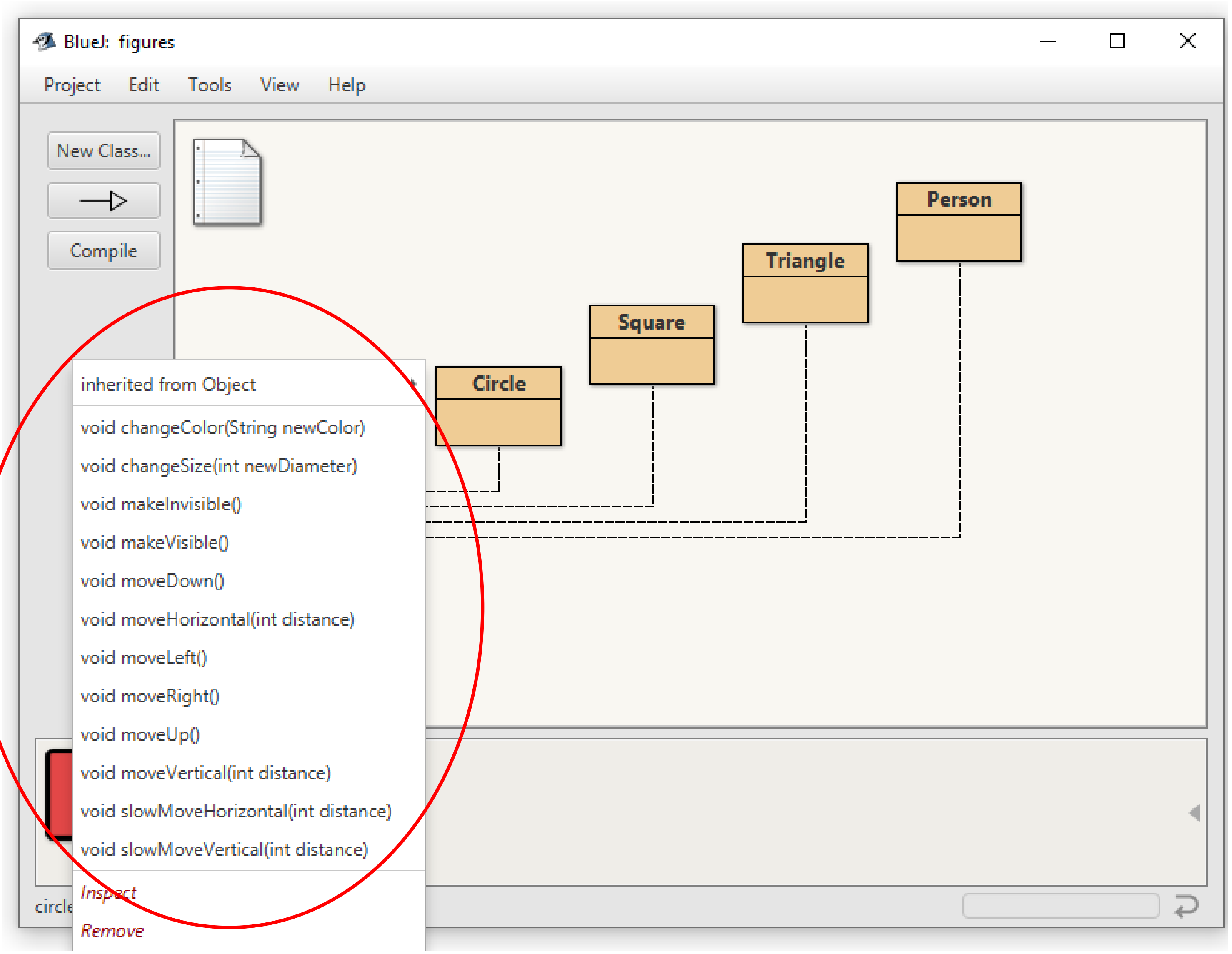
Methods and parameters for a Circle object

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



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

Methods and parameters for a Circle object



inherited from Object
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)
Inspect
Remove

Method signatures

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

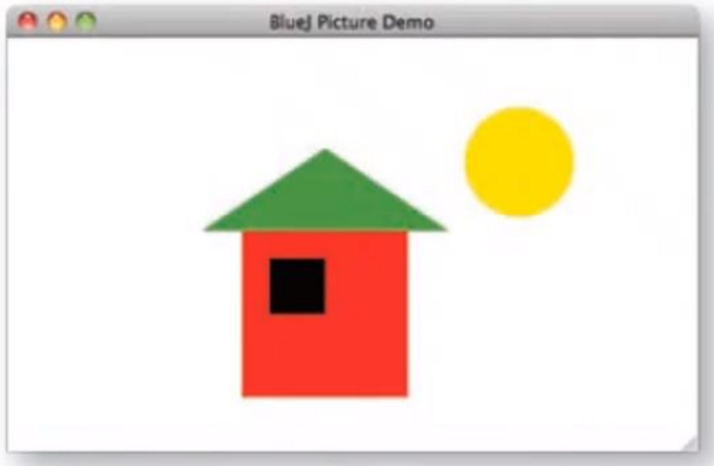
- The name of a method, together with the types of the parameters, are called the **signature** of the method.
- The method signature provides the necessary information to invoke the method.
- Q: What is the signature of the changeSize method?
- Q: What are the differences between the signatures of the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

More on methods

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

- Parameters pass additional information needed to execute a method. They act as “input” to the method.
- Parameters have types.
- Methods may also return a *result* via a return value.
  - All the methods above have “void” return type, indicating that they “do” things, rather than returning information.
- Objects **communicate** by calling each other’s methods.

Exercise



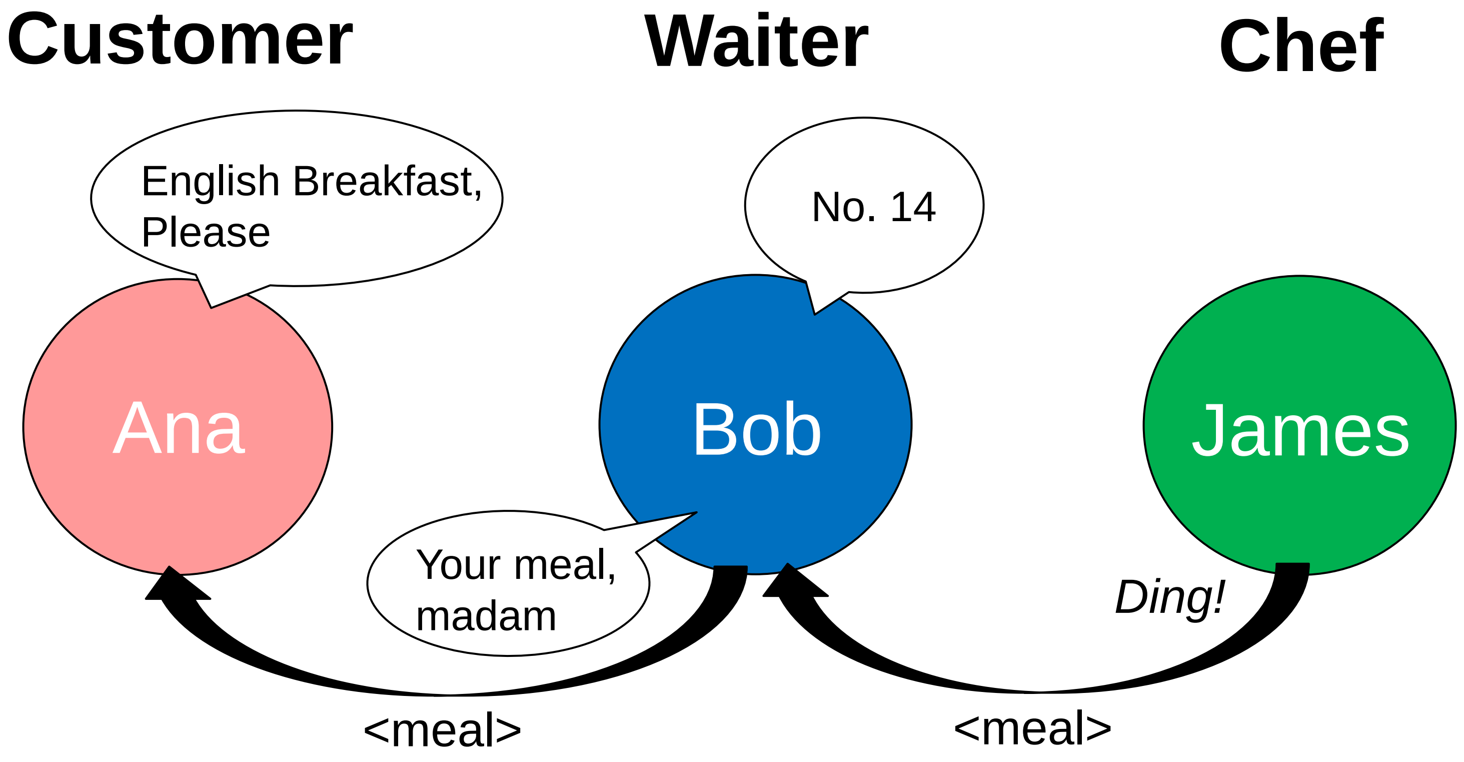
Exercise 1.9. - Recreate the image shown above using the shapes from the ***figures*** project. While you are doing this, write down what you have to do to achieve this. Could it be done in different ways?

Example - Restaurant

- Writing programs is largely about managing complexity
- How is something complex organized in the real world?
- Consider a restaurant:
  - Customers order meals
  - Chefs prepare the dishes
  - Waiters take orders, and bring food to the tables
  - Barmen prepare and serve drinks

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

Example - A Restaurant



Example - Restaurant

- In this scenario, a Waiter has the following *actions* that it can perform:
  - Bring menus
  - Take orders
  - Bring meals

As a costumer we can deal with any individual waiter, based solely on our knowledge of what things a Waiter can do.

Recap

- Now you should be able to give an explanation of each of these terms:
  - Object
  - Class
  - Method
  - Parameter
  - Signature
  - Type
  - State

Next time: Understanding Class Definitions (Chapter 2)