Java I

Objects & Variables

Static Functions

Last week we wrote our first Java functions to calculate pet human ages.

That was very procedural, not very OO - or Object Oriented.

That's not very Java… Today we'll fix that.

Classes & Objects

Let’s start with questions…

- What is a class?
- What is an object?
- Are they the same thing?

A class is a pattern

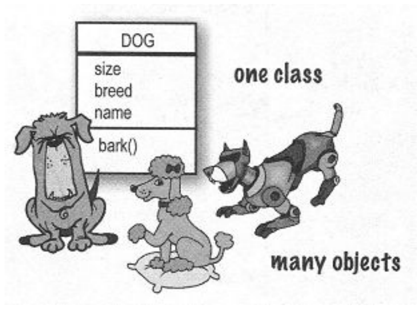
A class is what you use to make an object.

It’s like a pattern or a blueprint.

Sometimes I’ll describe it as a cookie-cutter.

Dog class

For example:
You might have a Dog class and then use it to make many different Dog objects



Each object is an instance

- A class is the definition of an object
  - the pattern or blueprint
- An instance is an object created from it.
  - When you are designing, you tend to say object.
  - When you are coding, you tend to say instance.
- The class is the cookie cutter, the instances the cookies

Object state & behavior

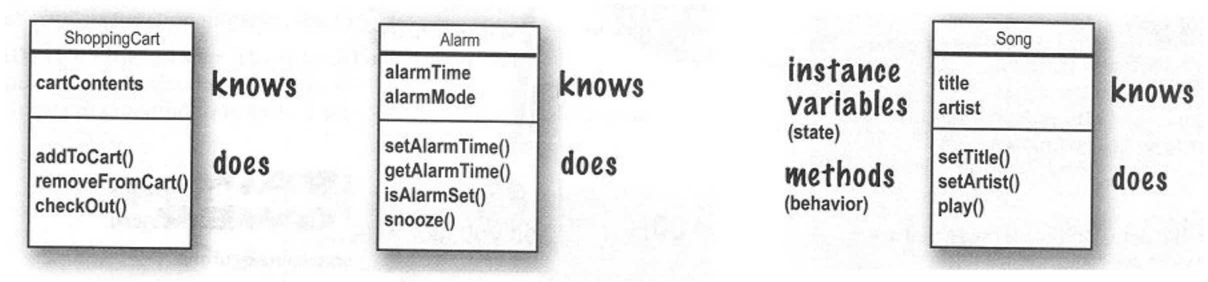
State is the object's data. Things it "knows".

- These become instance variables

Behavior is what the object can do.

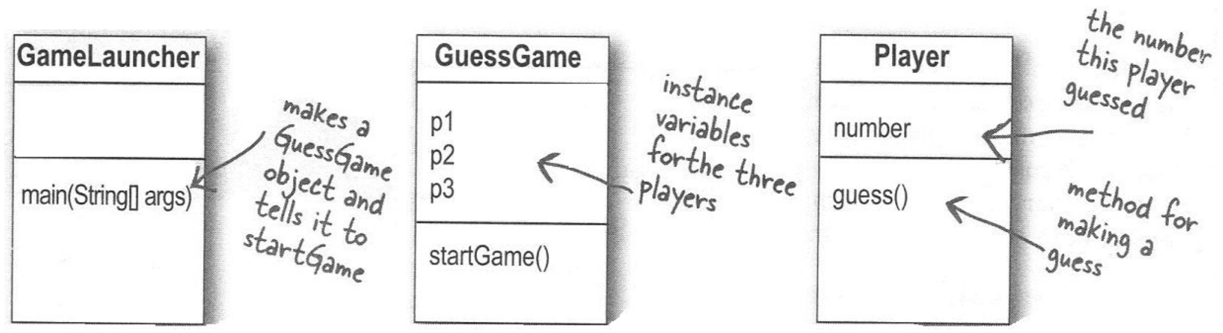
- These become methods

Object state & behavior



Java applications

Java applications are made of many classes.


GameLauncher
main(String[] args)
makes a GuessGame object and tells it to startGame
GuessGame
p1
p2
p3
startGame()
instance variables forthe three players
Player
number
guess()
the number this player guessed
method for making a guess

“Main” class

Usually there is one “main” class that contains the `main` method and little else.

That main method will create one or more instances of other classes and call methods on them.

Object review

- All java code is defined in a class
- A class describes how to make an instance of an object
- An object has state (instance variables) and behavior (methods)
- Applications are made of many classes
- One main class to run the program

Reference variables

- Reference variables are used to access an object.
- It's like a remote control that you use to work with the actual object.

Reference variables

In Java, exactly what this reference variable is doesn't matter.

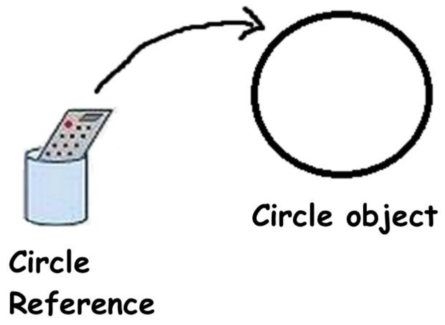
We just know that whatever it is:

- it represents exactly one object
- the JVM knows how to use it

Reference variable type

Reference variables still have types.

If we make a (reference) variable of type Circle, it will refer to an object with the type of Circle.



Primitives vs References

- Primitive values are stored right inside of the primitive variables.
- Reference variables hold bits that refer to an object.

Arrays

Declare an int array variable:

```
int[] numbers;
```

Create a new int array with length of 7 & assign it to the variable:

```
numbers = new int[7];
```

Arrays

Give an element in the array a value:

```
numbers[0] = 6;    //the first position is 0 not 1
numbers[1] = 23;
```

Alternate syntax to create and initialize:

```
int[] numbers = new int[] {6, 23,
31, 18};
```

Arrays are objects

An array is an object.

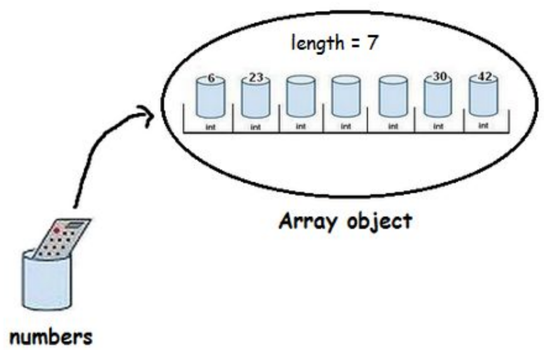
One of the properties of an array is it's length (or size).

Arrays are objects

- Because it is an object, it lives on the heap.
- The amount of space it needs is based on it’s size, which is set when you create it.
- An array’s size cannot be changed.

Arrays are objects

You can think of it like a tray of cups…



Array length

- length is a property of the array object.
- You access properties (if they are visible) using the dot (.) operator.

To get the length of our array:

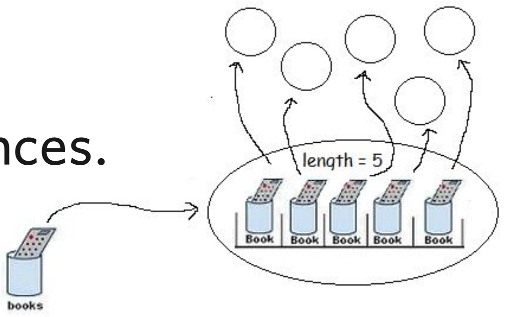
```
int arraySize = numbers.length;
```

Object arrays are similar

Arrays can also hold objects.

```
Book[] books = new Book[5];
```

Instead of primitive values
in the cups,you have references.



Object arrays are similar

To access specific cups use with [] with the position:

```
Book myBook = books[5];
```

If you ask for a position greater than or equal to the length,
you will get an exception (a type of error).

Variables Review

Variables are either primitive or reference.

Variables always have a name and a type.

A primitive variable holds bits representing it's value.

A reference variable holds bits that refer to an object.

Variables Review

Objects live on the heap.

An array is always an object, even if it holds primitive types.

Access object properties with the dot (.) operator.