GET SMART: WITH JAVA PROGRAMMING

YAMAN OMAR ALASHQAR

SYSTEM.OUT.PRINTLN(“WELCOME TO THIS COURSE”);

INTRODUCTION TO CLASSES

All programs require one or more classes that act as a model for the world.

CLASSES

Class:
It is a template or blue print about the capability of what an object can do.

Object:
It’s the instance of a class/ it’s the working entity of a class.

Method:
The behaviors of a class. It tells what a object can do.

IMPLEMENTING A CLASS

CREATING AN INSTANCE

ACCESSING OBJECT PROPERTIES

Access Modifier	Within Class	Within Package	Same Package by subclasses	Outside Package by subclasses	Global
Public	Yes	Yes	Yes	Yes	Yes
Protected	Yes	Yes	Yes	Yes	No
Default	Yes	Yes	Yes	No	No
Private	Yes	No	No	No	No

- Real Life Objects, which has some Properties, and Methods
- In real life, a car is an object.
- A car has properties like weight and color, and methods like start and stop

Properties	Methods
car.name = Fiat	car.start()
car.model = 500	car.drive()
car.weight = 850kg	car.brake()
car.color = white	car.stop()

- All cars have the same properties, but the property values differ from car to car.
- All cars have the same methods, but they are performed at different times.
- Methods are actions that can be performed on objects


remote1.volumeUp()
remote1.play()
r2.play()

- A method is a block of code, designed to perform a particular task

- Why methods?
  - You can reuse code: Define the code once, and use it many times.
  - You can use the same code many times with different arguments, to produce different results

- arguments are the values received by the function when it is invoked

- Inside the method, the arguments (parameters) behave as local variables
- Variables declared within a method, become LOCAL to the function.
- Local variables can only be accessed from within the function.

- Declared methods are not executed immediately.
  - They are "saved for later use",
  - and will be executed later, when they are invoked (called upon)

- When Java reaches a return statement, the method will stop executing
- If the method was invoked (called) from a statement, Java will "return" to execute the code after the invoking statement.
- Method’s often compute a return value.
    - The return value is "returned" back to the "caller“
- In a method, this refers to the "owner" of the function

CONSTRUCTORS

A constructor in Java is a special method that is used to initialize objects.
The constructor is called when an object of a class is created.
It can be used to set initial values for object attributes

Note that the constructor name must **match the class name**,
and it cannot have a **return type**

Also note that the constructor is called when the object is created.

All classes have constructors by default:
if you do not create a class constructor yourself, Java creates one for you.

Constructors can also take parameters, which is used to initialize attributes

SQUARE

Java is pass-by-value.

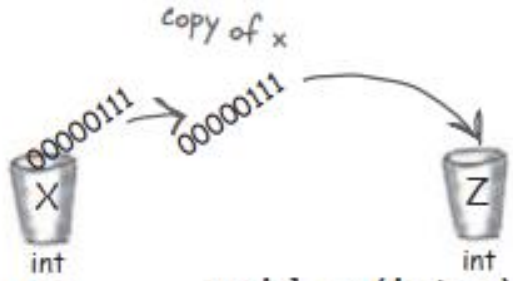
That means pass-by-copy.

```
int x = 7;
```

1 Declare an int variable and assign it the value '7'. The bit pattern for 7 goes into the variable named x.

```
void go(int z){ }
```

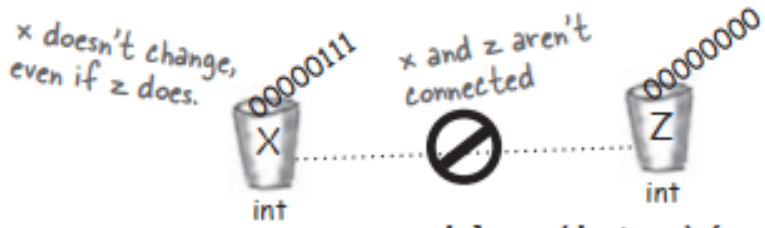
2 Declare a method with an int parameter named z.




```
foo.go(x);
```

```
void go(int z){ }
```

3 Call the go() method, passing the variable x as the argument. The bits in x are copied, and the copy lands in z.




```
void go(int z){
    z = 0;
}
```

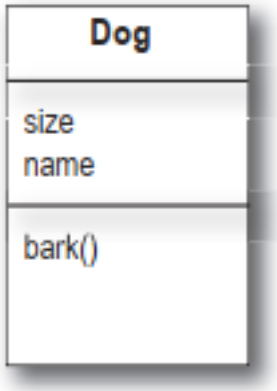
4 Change the value of z inside the method. The value of x doesn't change! The argument passed to the z parameter was only a copy of x.

The method can't change the bits that were in the calling variable x.

The size affects the bark

A small Dog's bark is different from a big Dog's bark.

The Dog class has an instance variable *size*, that the *bark()* method uses to decide what kind of bark sound to make.




```
class Dog {
  int size;
  String name;

  void bark() {
    if (size > 60) {
       System.out.println("Wooof! Wooof!");
    } else if (size > 14) {
       System.out.println("Ruff!  Ruff!");
    } else {
       System.out.println("Yip! Yip!");
    }
  }
}
```

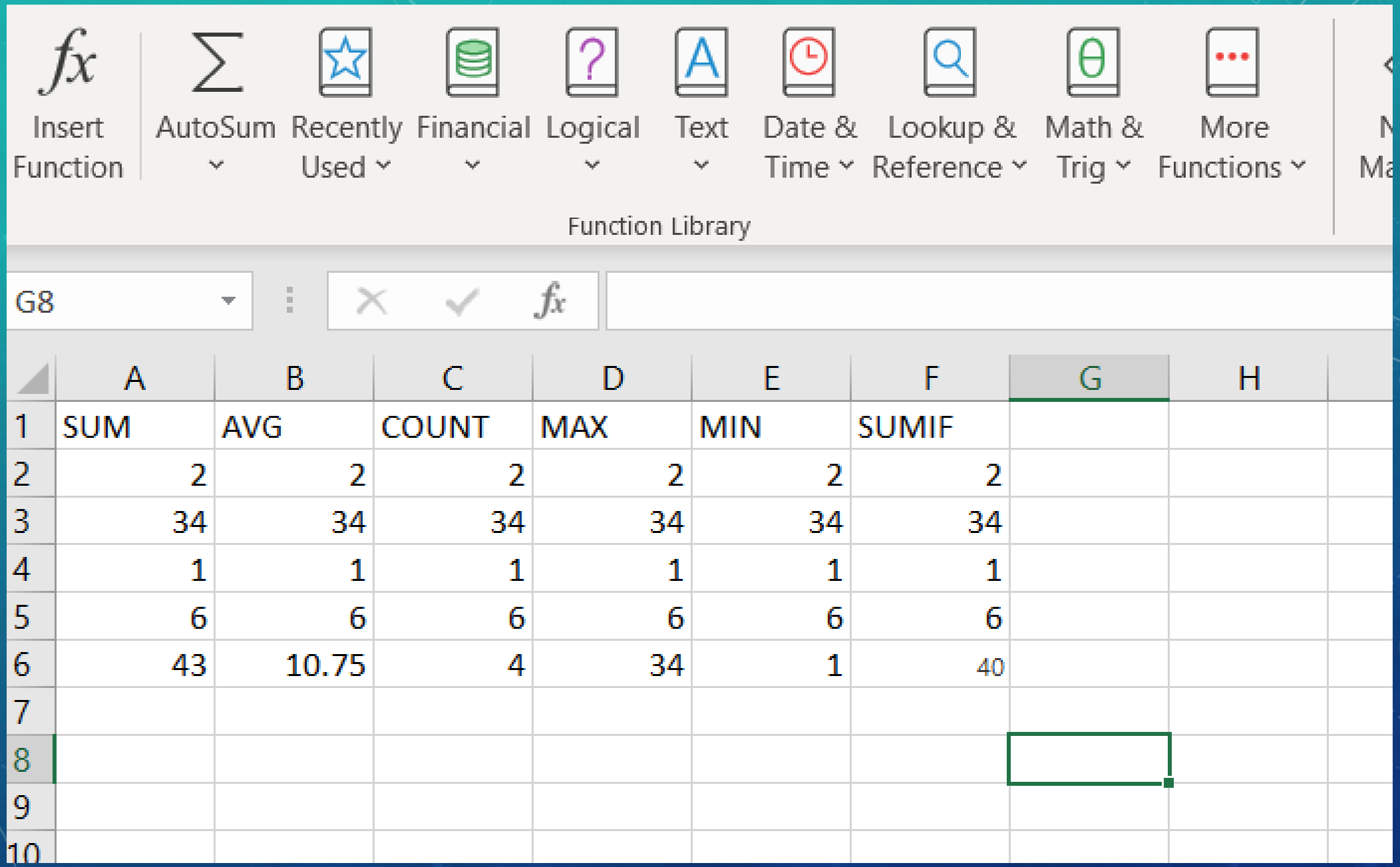
```
class DogTestDrive {

   public static void main (String[] args) {
     Dog one = new Dog();
     one.size = 70;
     Dog two = new Dog();
     two.size = 8;
     Dog three = new Dog();
     three.size = 35;

     one.bark();
     two.bark();
     three.bark();
   }
}
```

```
File Edit Window Help Playdead
%java DogTestDrive
Wooof! Wooof!
Yip! Yip!
Ruff!  Ruff!
```

Exercise: Make each dog bark 3 times

- Classes define what an object knows and what an object does.
- Things an object knows are its **instance variables** (state).
- Things an object does are its **methods** (behavior).
- Methods can use instance variables so that objects of the same type can behave differently.
- A method can have parameters, which means you can pass one or more values in to the method.
- The number and type of values you pass in must match the order and type of the parameters declared by the method.
- Values passed in and out of methods can be implicitly promoted to a larger type or explicitly cast to a smaller type.
- The value you pass as an argument to a method can be a literal value (2, 'c', etc.) or a variable of the declared parameter type (for example, *x* where *x* is an int variable). (There are other things you can pass as arguments, but we're not there yet.)
- A method *must* declare a return type. A void return type means the method doesn't return anything.
- If a method declares a non-void return type, it *must* return a value compatible with the declared return type

- Write a method that returns the largest element in a list/array
- Write a method that checks whether an element occurs in a list/array
- Write a method that returns the sum of odd positions in a list
- Write a method to return all odd values in a list of int
- Write a method that concatenates two lists
- Write a method that combines two lists by alternatingly taking elements,
    - e.g. [a,b,c], [x,y,z] → [a,x,b,y,c,z]
- Write a method that takes a number and returns a array of its digits.
    - So for 2342 it should return [0,0,2,3,4,2]
    - and for 12345 return [0,1,2,3,4,5]
    - and for 25 return [0,0,0,0,2,5]



	A	B	C	D	E	F	G	H
1	SUM	AVG	COUNT	MAX	MIN	SUMIF		
2	2	2	2	2	2	2		
3	34	34	34	34	34	34		
4	1	1	1	1	1	1		
5	6	6	6	6	6	6		
6	43	10.75	4	34	1	40		
7								
8								
9								
10								