ICS 141 Programming with Objects

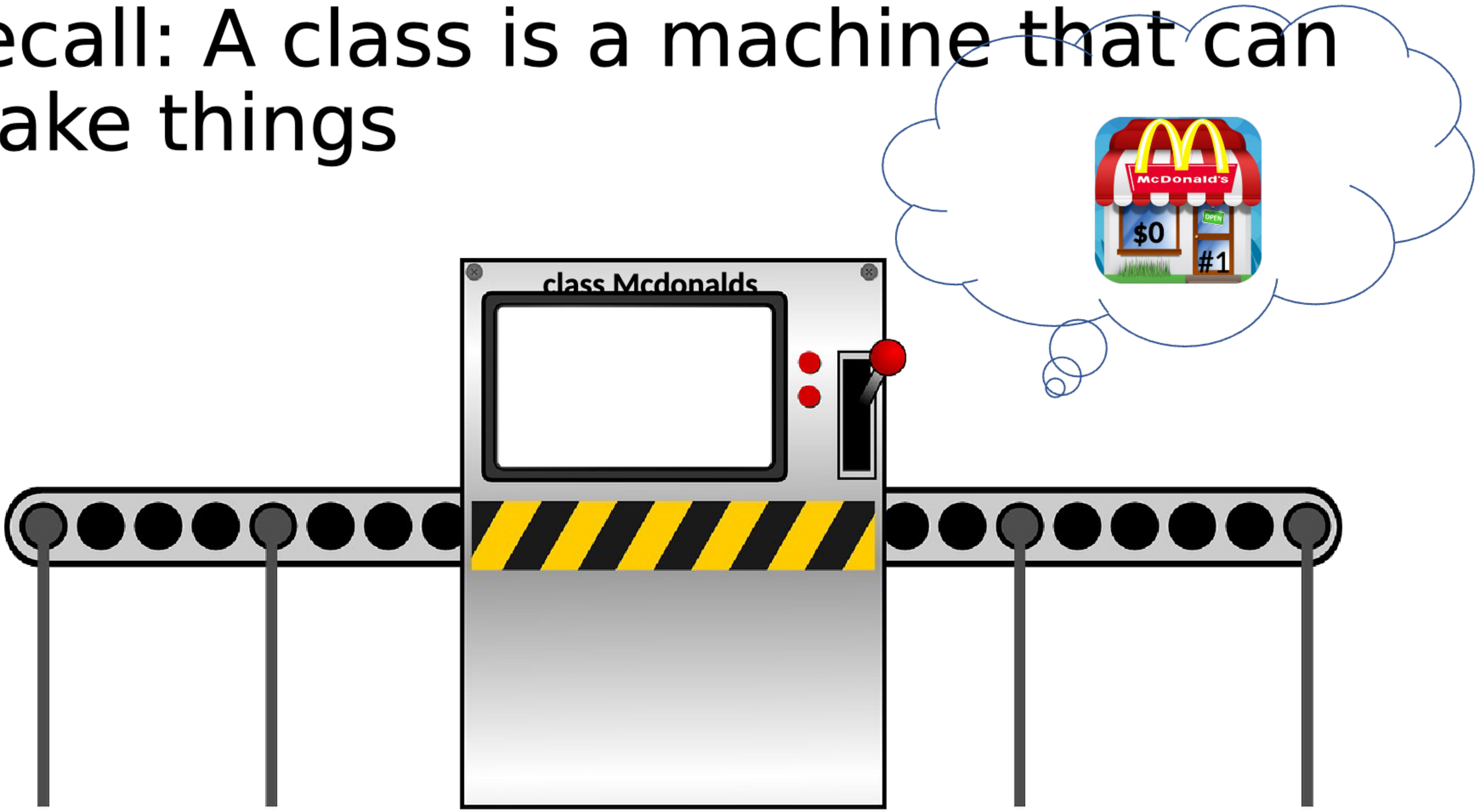
Jessica Maistrovich

Metropolitan State University

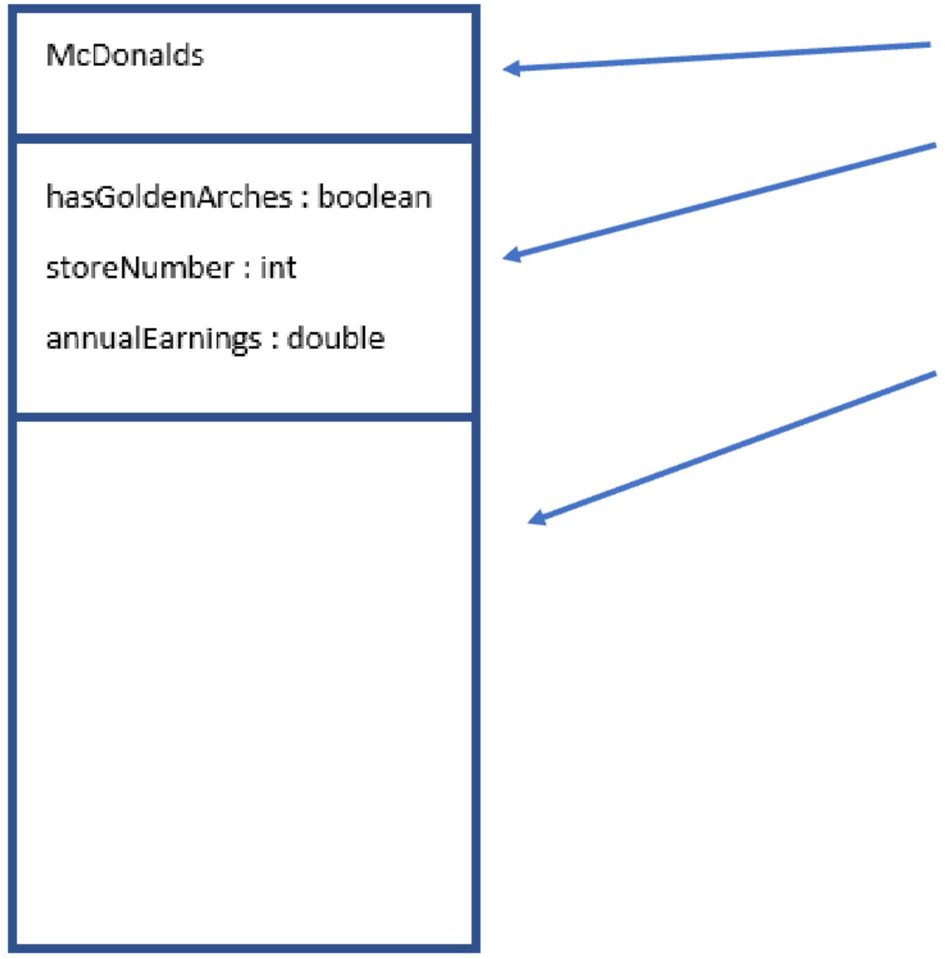
Classes – Part 2

Review what we know so far

Recall: A class is a machine that can make things



Recall UML Diagram



Noun (object)

Adjectives (instance variables)

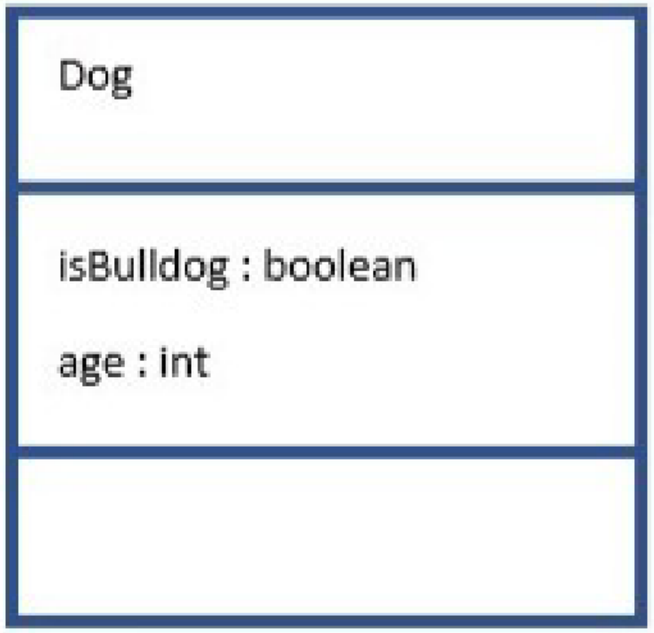
Describe the state of the object

Verbs (methods)

Describe what the object can do

Try it!

- Open Eclipse and create a project called DogApplication
- Create a class based on the following UML:



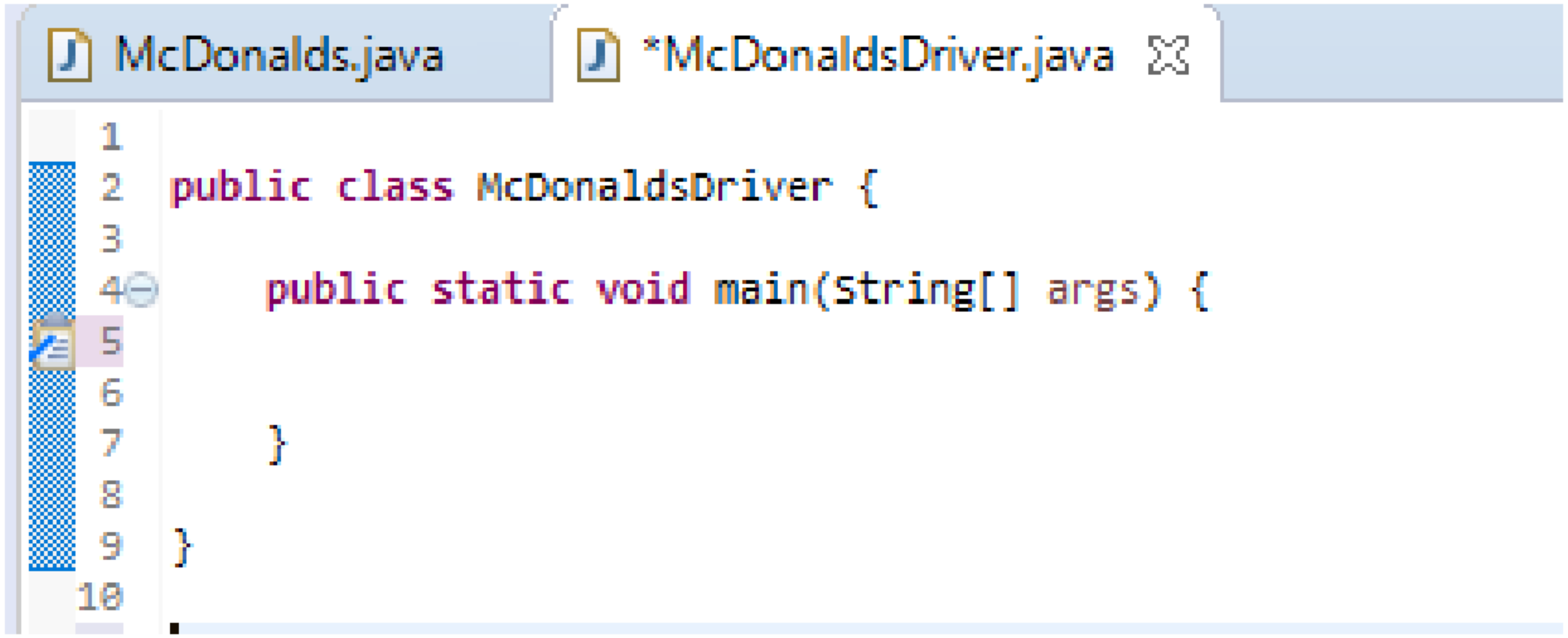
Driver class

Has main method.

Boss class

- Runs everything. Pushes buttons on machines to make objects
- Usually boss doesn’t do any “work” – just tells everyone else to
- Don’t use word boss – instead use word driver

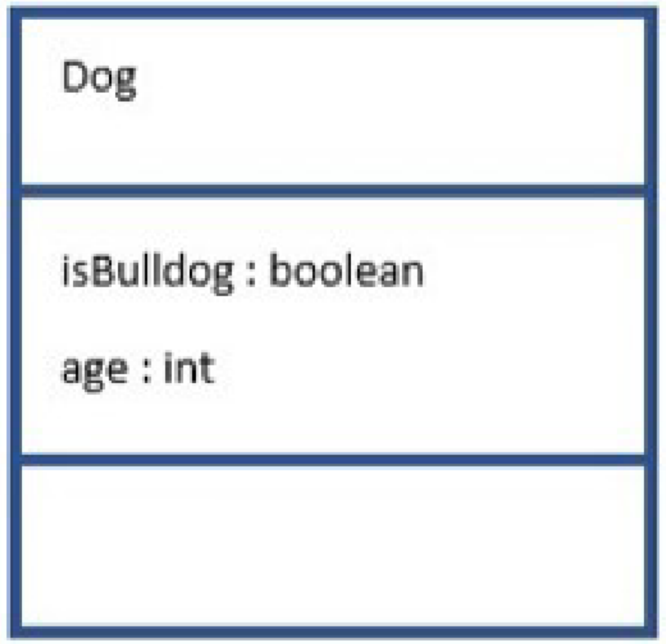
- Could have main method inside small classes
  - One person organization – serve multiple roles

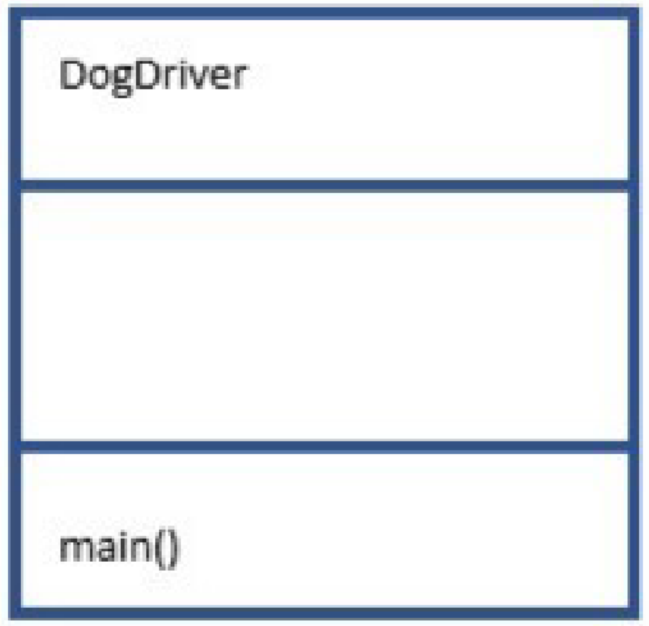


```java

public class McDonaldsDriver {

    public static void main(String[] args) {


    }

}

```

Try it!

- Add a driver class to the DogApplication project

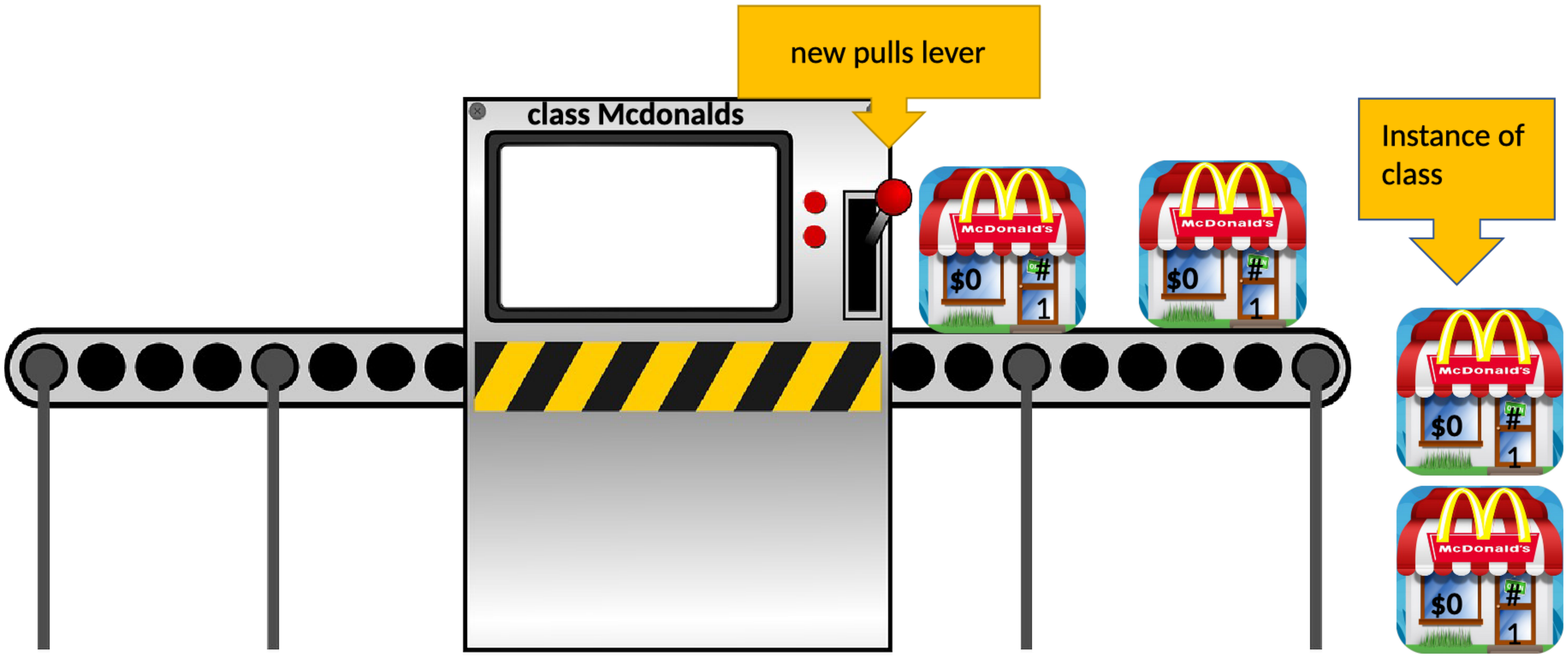


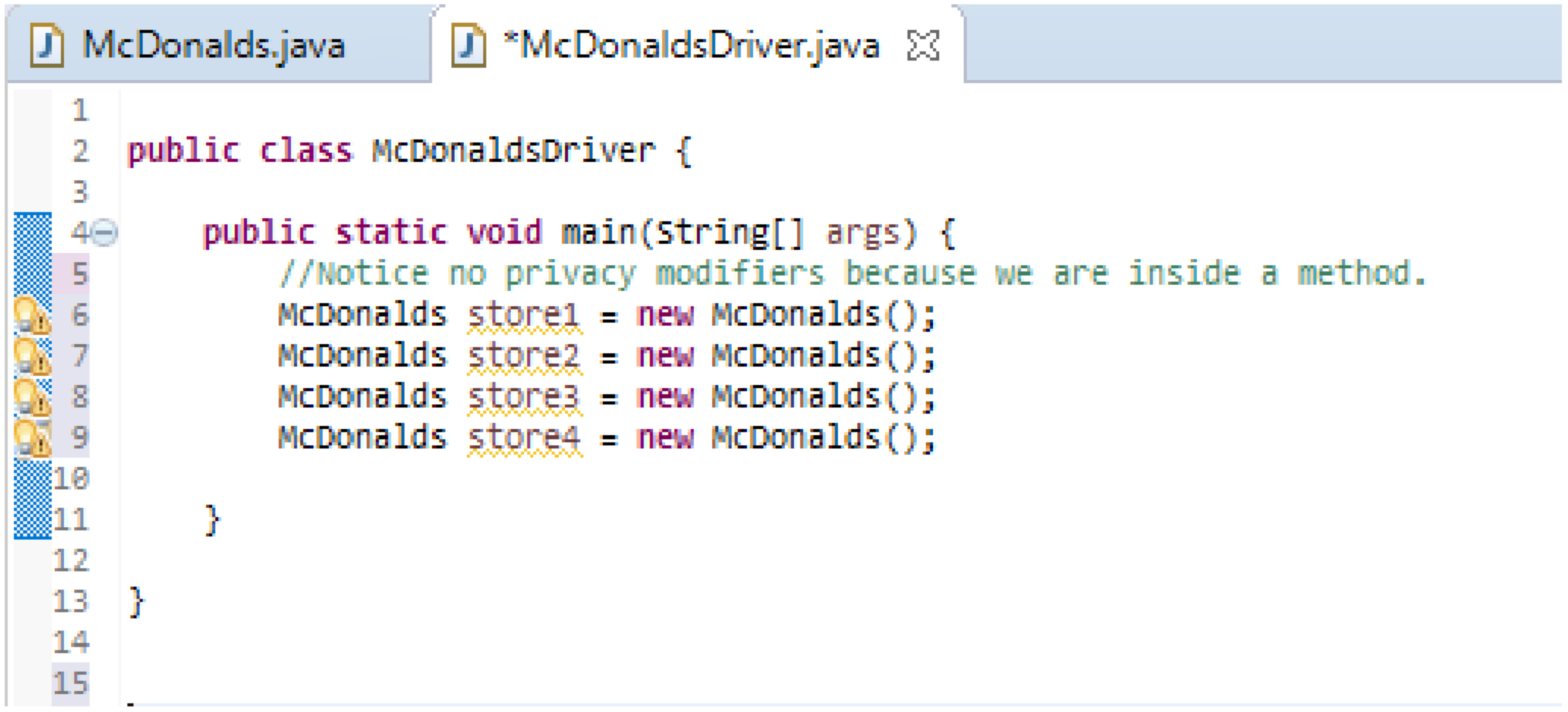


Instantiating an Object

Using “new”

Use new to instantiate




McDonalds.java
*McDonaldsDriver.java

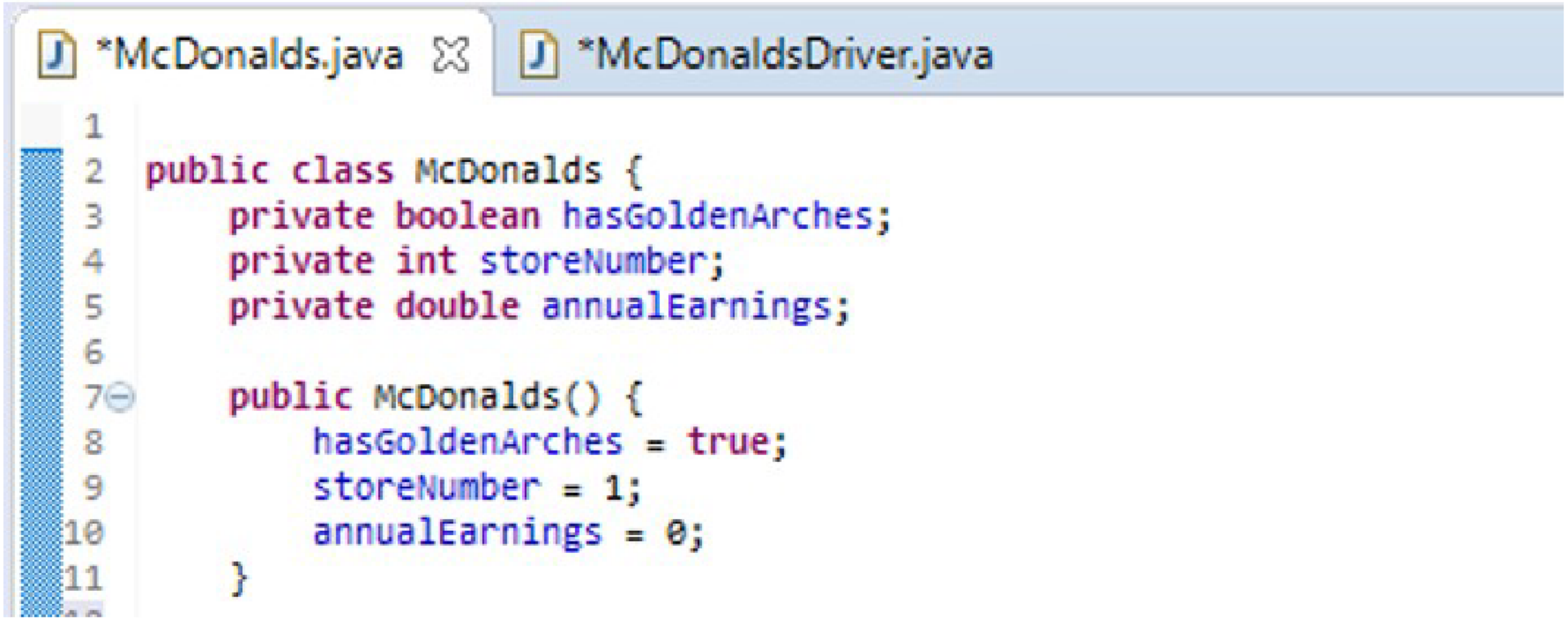
```java

public class McDonaldsDriver {

    public static void main(String[] args) {
        //Notice no privacy modifiers because we are inside a method.
        McDonalds store1 = new McDonalds();
        McDonalds store2 = new McDonalds();
        McDonalds store3 = new McDonalds();
        McDonalds store4 = new McDonalds();

    }

}

```

Constructors

A constructor helps you “build” the object

Constructor (Assign values to variables)

- Usually give initial values to variables (initialize) with a special method called a constructor (can tell it is a constructor because it has the same name as the class and no return type)
  - Not ideal to use methods (including setters and getters) in constructor
  - Assignment occurs with =
- The constructor is called during instantiation
- No-argument (no-arg) constructor has no parameters
- Default constructor is a no-arg constructor that sets everything to “nothing”
  - Once you code a constructor, there is no longer a default constructor



*McDonalds.java   *McDonaldsDriver.java

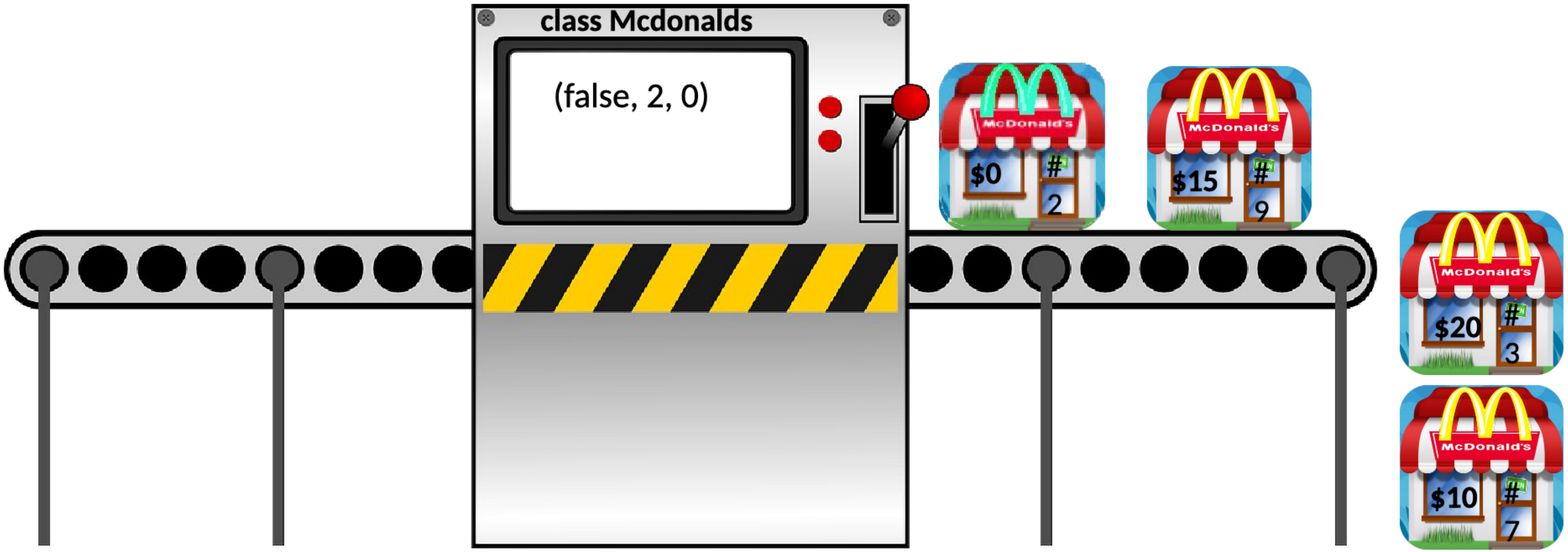
```java

public class McDonalds {
    private boolean hasGoldenArches;
    private int storeNumber;
    private double annualEarnings;

    public McDonalds() {
        hasGoldenArches = true;
        storeNumber = 1;
        annualEarnings = 0;
    }
```

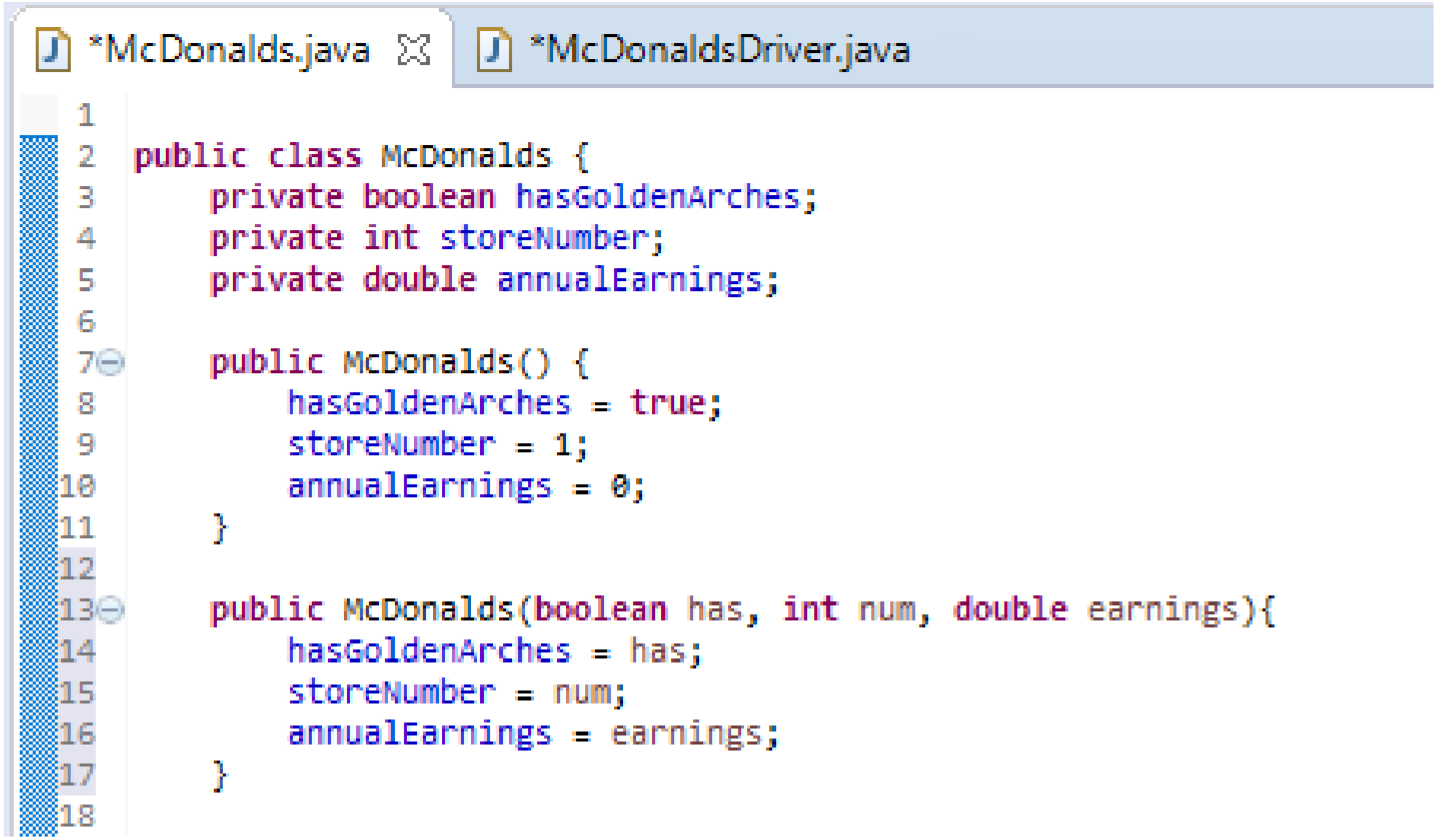
Constructors that accept values / Overloading methods

- Want to be able to initialize with different values
  - McDonalds in Sedona, AZ with teal arches

- Methods can have same name as long as they accept different parameters.
  - public McDonalds ()
  - public McDonalds (int storeNum)
  - public McDonalds (boolean golden, int storeNum, double earnings)
  - If have above, can’t have public McDonalds (int earnings)
    - Java isn’t reading the variable name when searching for a match, just the variable type

Send inputs to the constructor





```java

public class McDonalds {
    private boolean hasGoldenArches;
    private int storeNumber;
    private double annualEarnings;

    public McDonalds() {
        hasGoldenArches = true;
        storeNumber = 1;
        annualEarnings = 0;
    }

    public McDonalds(boolean has, int num, double earnings){
        hasGoldenArches = has;
        storeNumber = num;
        annualEarnings = earnings;
    }

```

Try it!

1. Create a constructor in the Dog class that takes two input parameters and uses those parameters to initialize the instance variables.
2. In the main method inside the driver class, declare a variable of type Dog (call it myDog) and instantiate it with values of your choice.
3. In the main method inside the driver class, declare a variable of type Dog (call it yourDog). Instantiate it with appropriate values so that it represents a 2 year old bulldog.

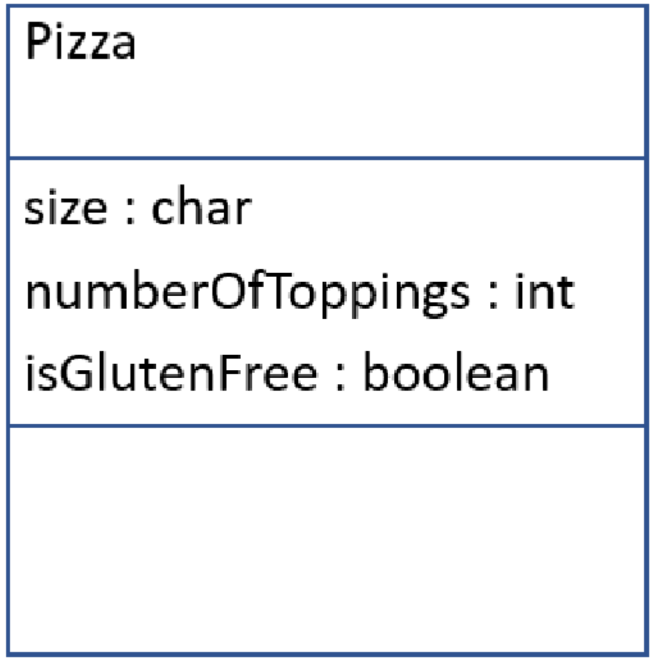
Accessors and Mutators

Official term for getters and setters

Great way to practice creating methods – everything is predetermined

Getters (Accessors)

- Returns the value that is stored in the variable
  - recall that we have hidden the information by using the private modifier
  - the getter “gets” or “allows access to” the information for others
- Always written as getVariableName()

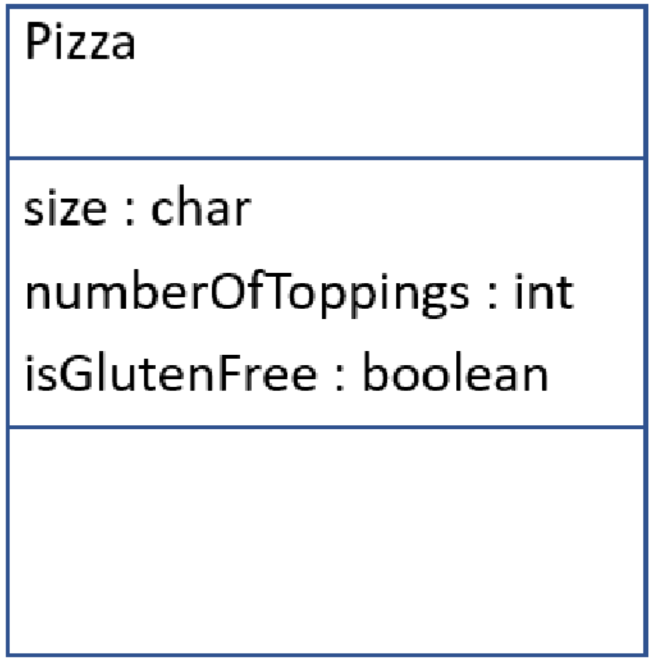


Add method to UML

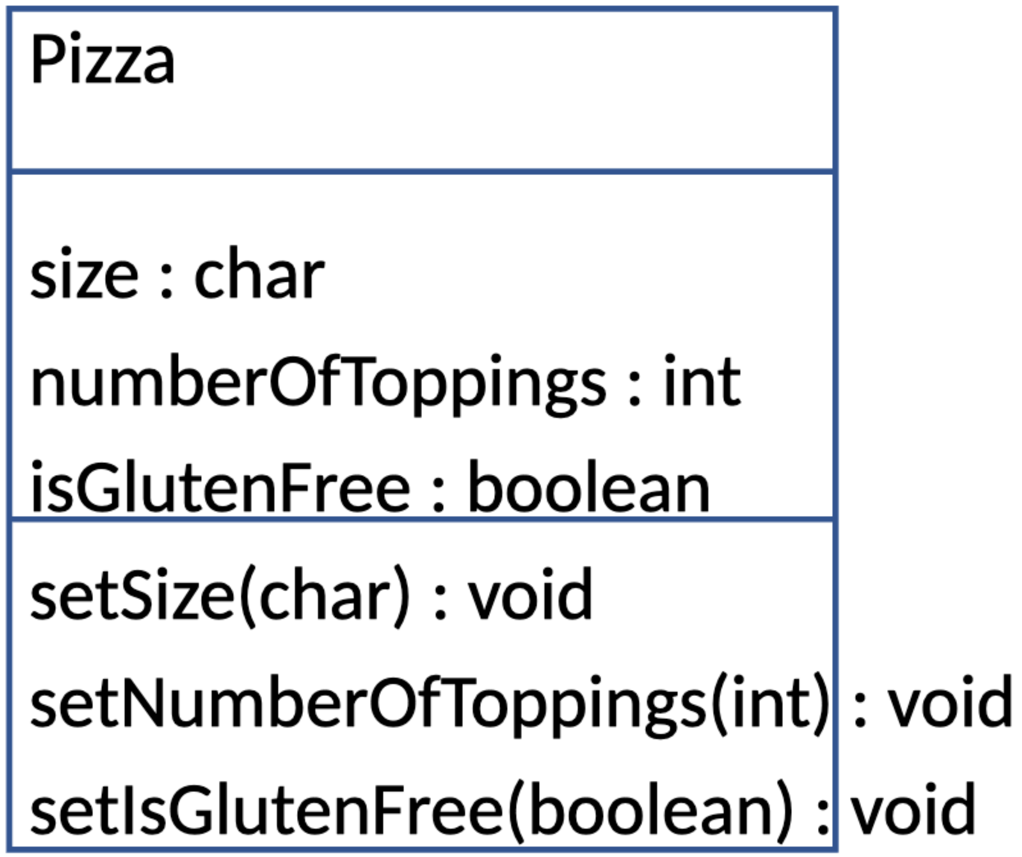
Pizza
size : char numberOfToppings : int isGlutenFree : boolean
getSize() : char getNumberOfToppings() : int getIsGlutenFree() : boolean

Setters

- Changes the value stored in the variable
  - the setter allows others to “set” or “mutate” the value stored in the variable
- Always written as setVariableName(parameter)



Add method to UML



Try it!

- Add a getter for all of the variables of the dog class.
- Add a setter for all of the variables of the dog class.

- Why is a getter useful? When should your class have a getter for a variable?

- Why is a setter useful? When should your class have a setter for a variable?