Gym App V1

New App (with Validation)

Produced by: Dr. Siobhán Drohan,
Ms. Mairead Meagher,
Ms. Siobhán Roche.

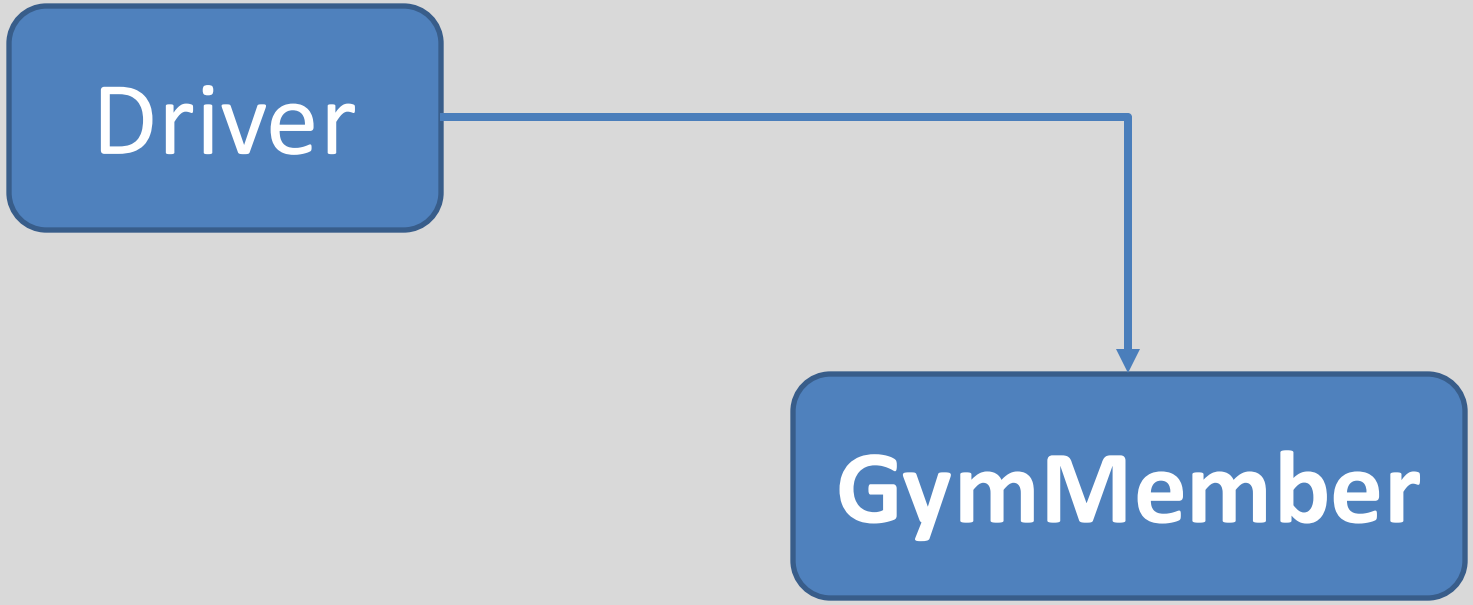
Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

Department of Computing and Mathematics
http://www.setu.ie/

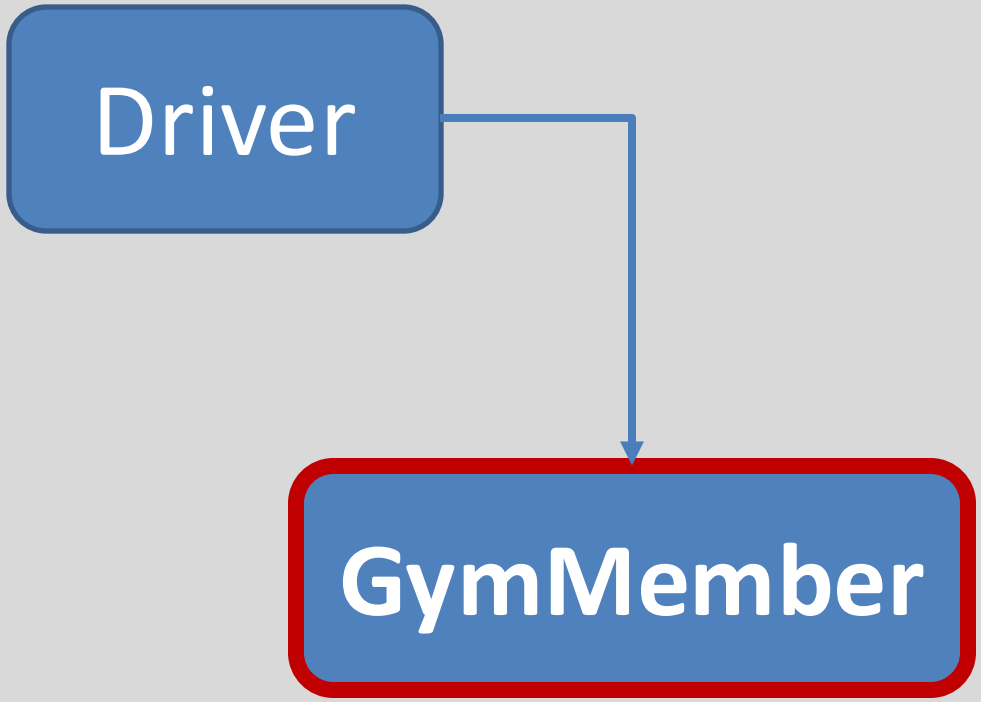
GYM APP

Version 1.0

Gym App V1.0



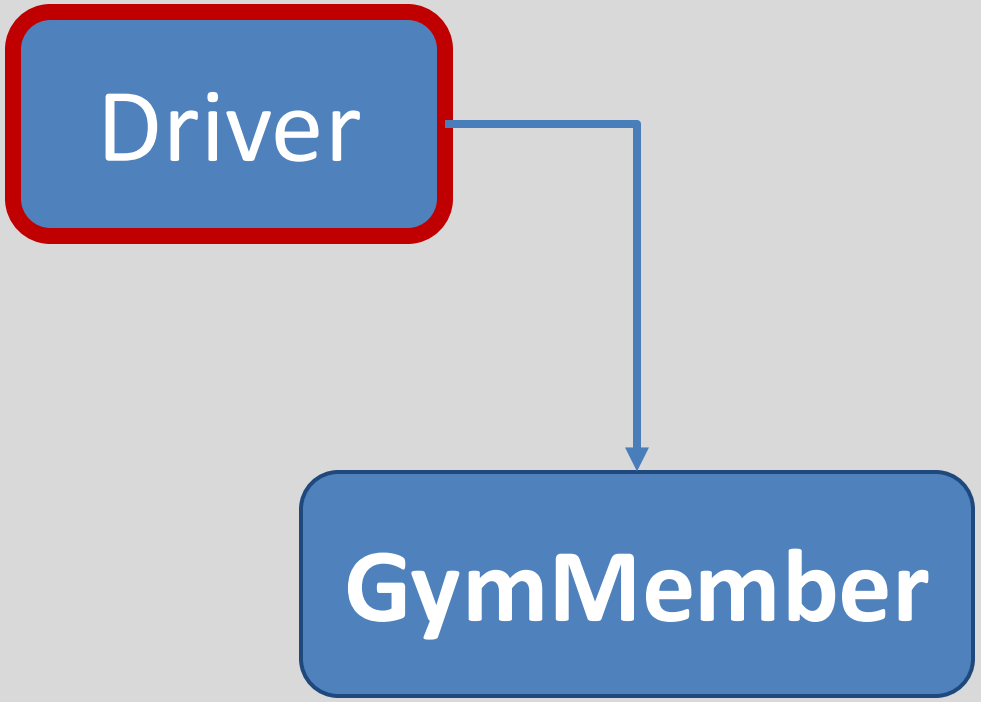
Gym App V1.0



The **GymMember** class stores **details** about a single member of a gym i.e.:

- name
- height
- weight
- membershipNumber
- if they are current members or not? (i.e. have they paid their membership fee)

Gym App V1.0



The **Driver** class:

- has the **main()** method.
- **reads** the gym member details from the user (via the console)
- **creates** a new GymMember object.
- **prints** the gym member details from the object (to the console)

Gym App V1.0 – Sample I/O

```
Driver.main({ });
Entering details
----------------
   Enter your name:                Dan Sheehan
   Enter your height (meters):     1.91
   Enter your weight (kgs):        111
   Enter the membership number:    222
   Is current member (y/n):        y


Printing details
----------------
Dan Sheehan: 1.91M, 111.0KG (Member Num: 222, current member: true)
```

ENCAPSULATED CLASS:

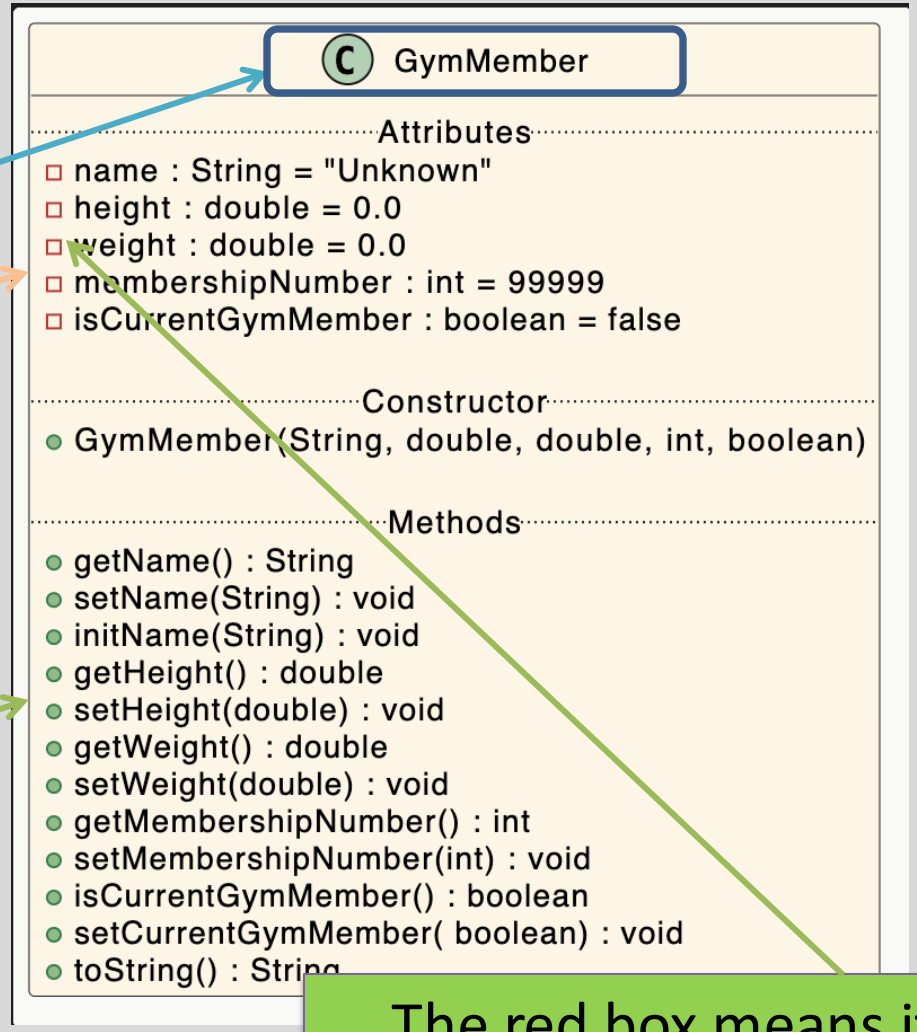
GymMember

A **GymMember** Class…

Object Type/ **Class** Name
i.e. GymMember

The **C** icon means it is a **Class**.

Fields/Attributes

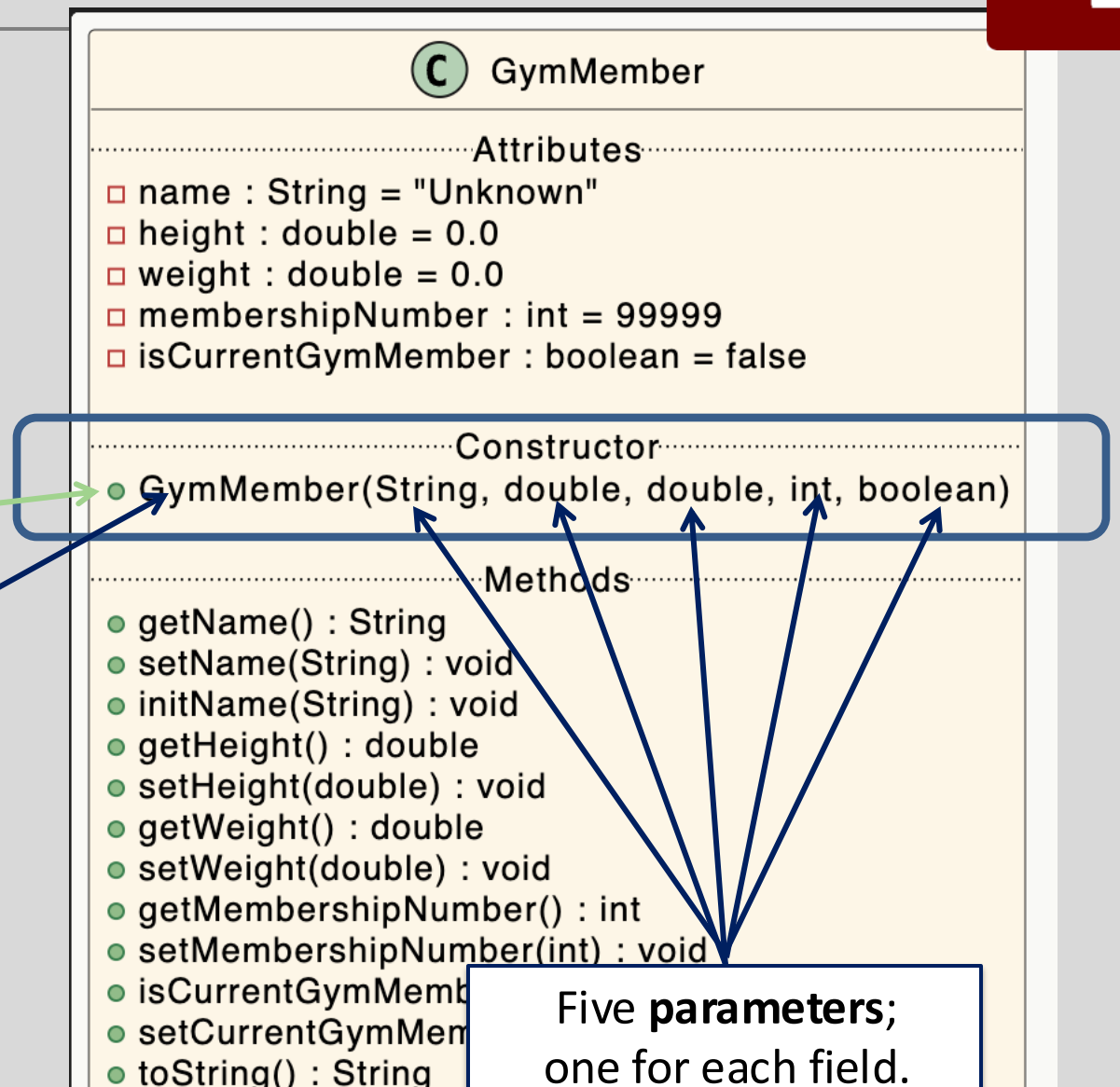


The green circle means it is **public**.

The red box means it is **private**.

A **GymMember** Class…Constructor

Constructor
i.e. for building objects.



The green circle means it is **public**.

Constructors have same name as the class

Five **parameters**;
one for each field.

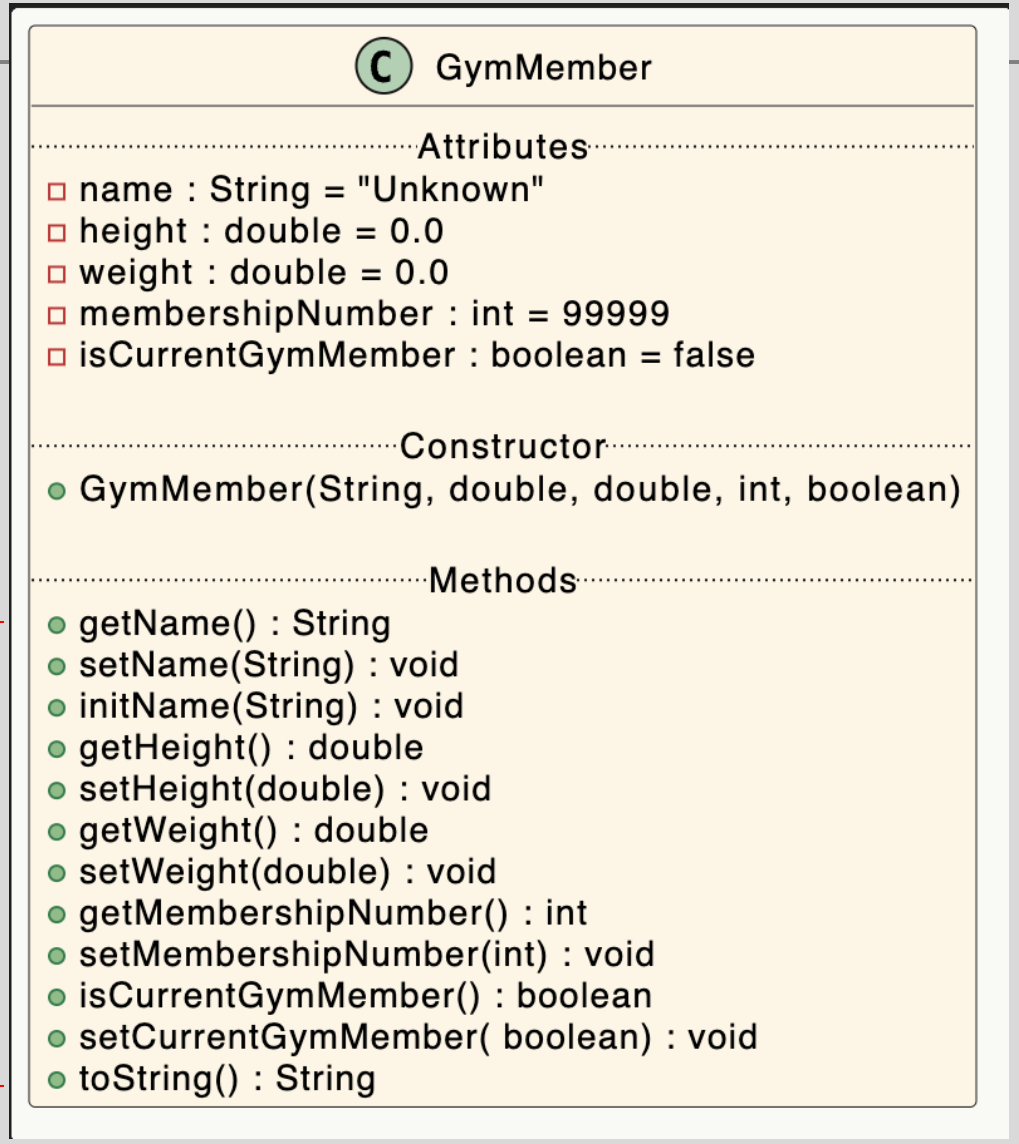
A **GymMember** Class…Fields and Constructor

```
private String name = "Unknown";
private double height = 0.0
private double weight = 0.0;
private int membershipNumber = 99999;
private boolean isCurrentGymMember = false;
```

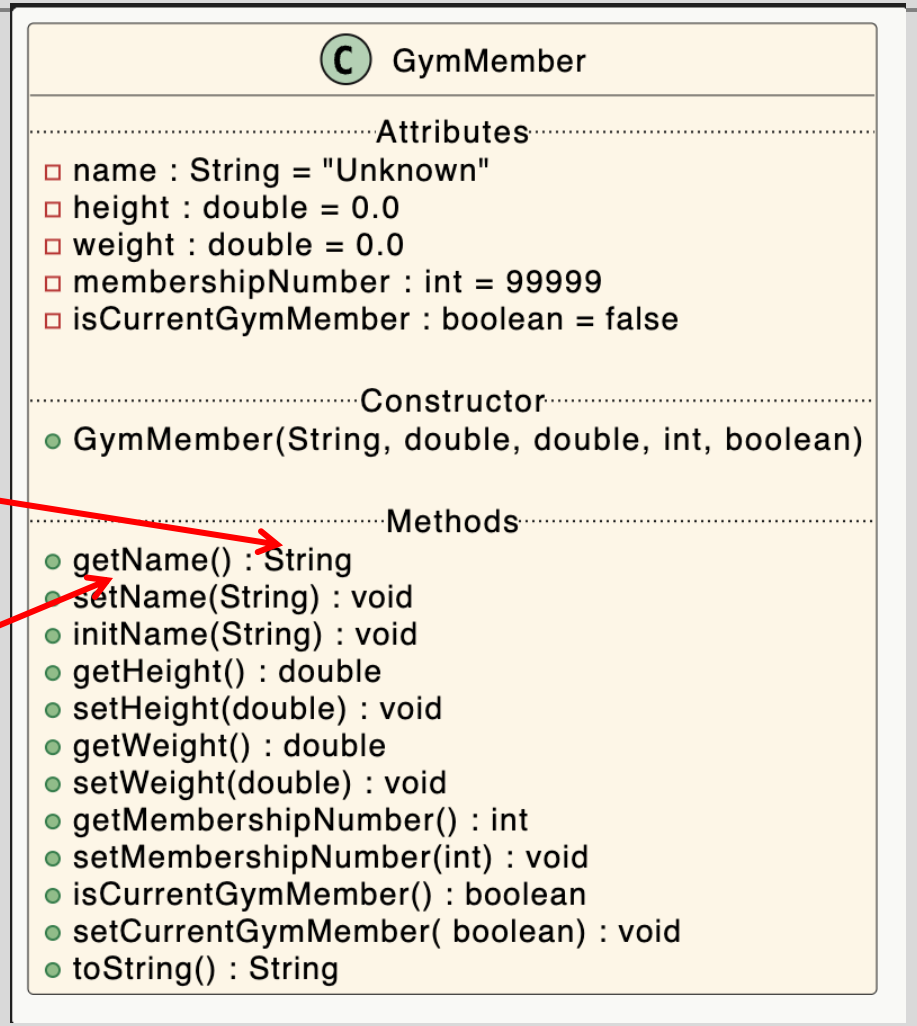
```
public GymMember(String name, double height, double weight, int membershipNumber, boolean isCurrentGymMember) {
    initName(name);
    setHeight(height);
    setWeight(weight);
    setMembershipNumber(membershipNumber);
    setCurrentGymMember(isCurrentGymMember);
}
```

A **GymMember** Class…methods

Methods
i.e. the **behaviours** of the class



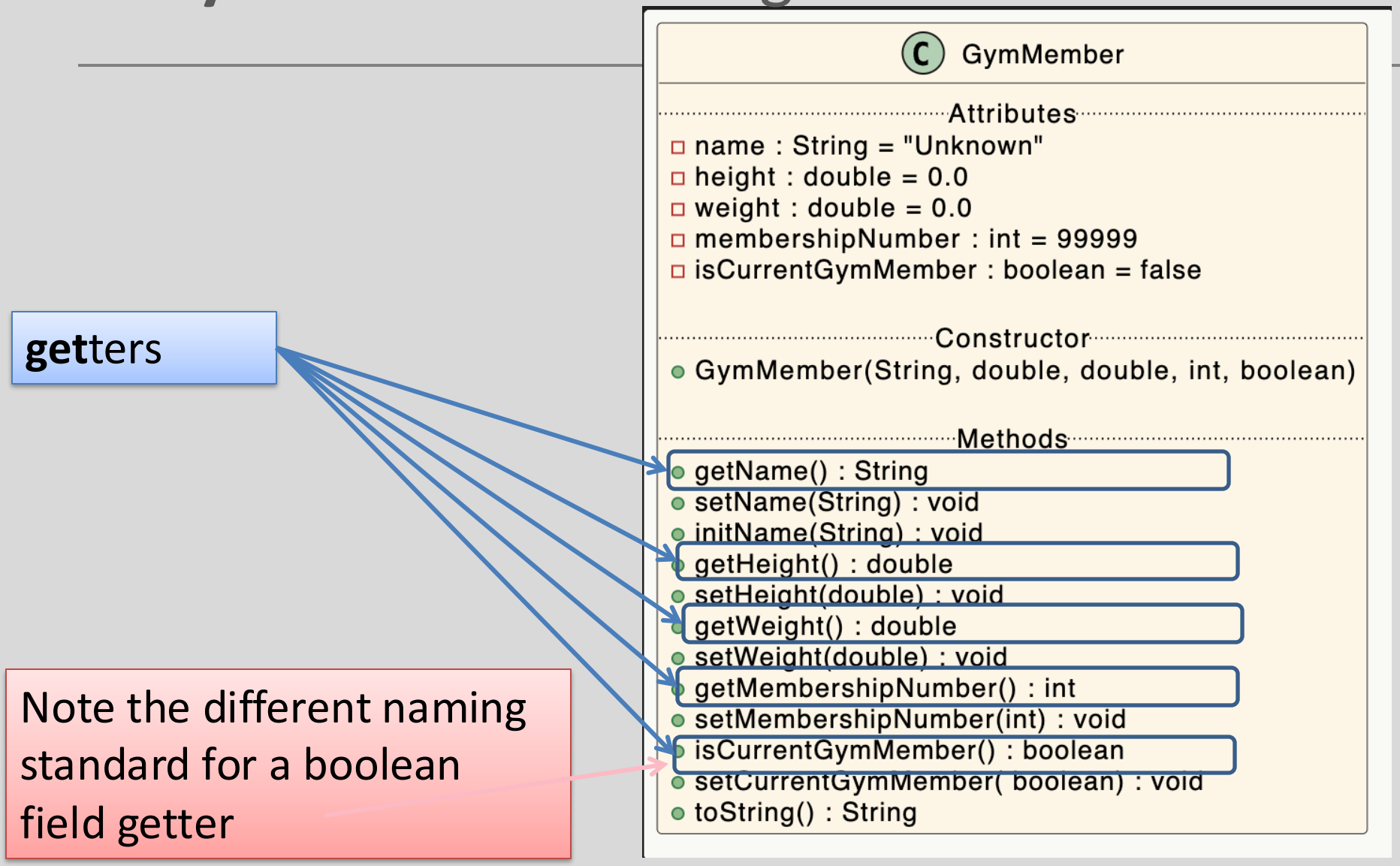
A **GymMember** Class…methods



Return type

Method **name**

A **GymMember** Class…getters

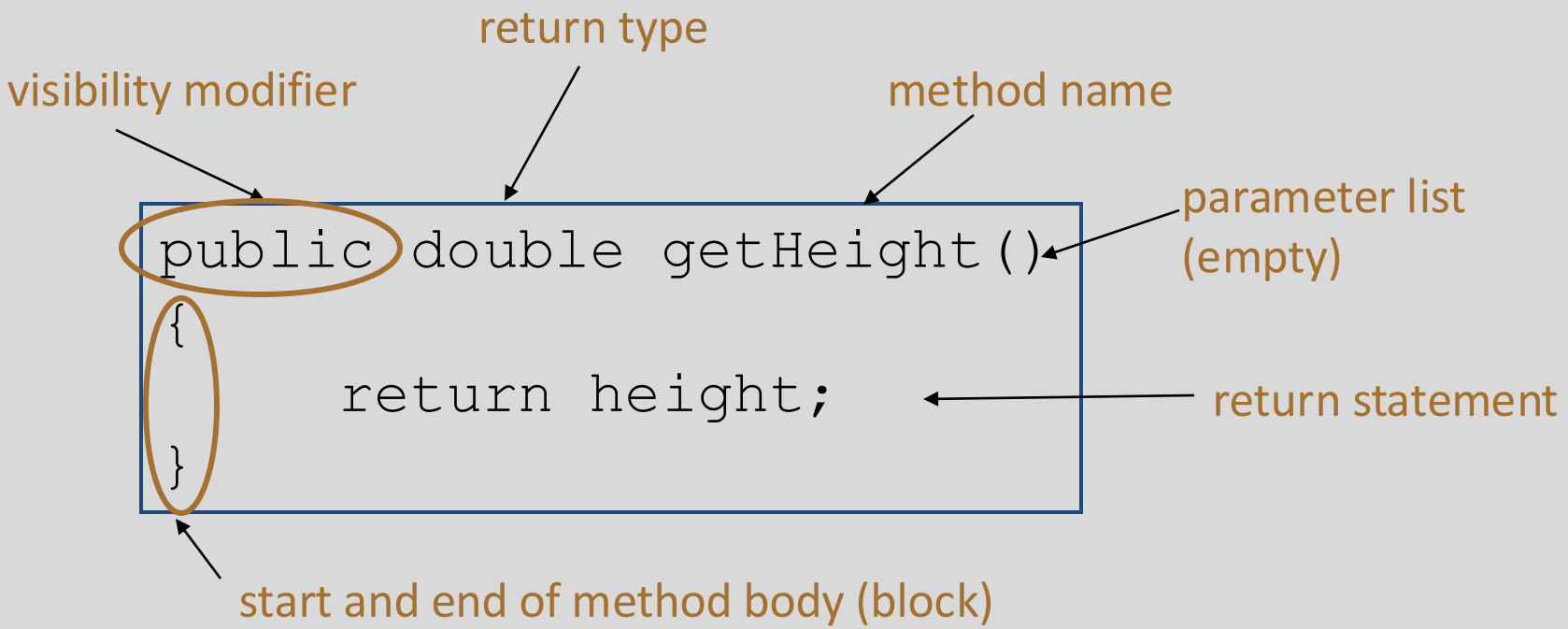


Getters (Accessor Methods)

- **Accessor** methods
  - return information about the **state** of an object
    - i.e. **the values stored in the fields**.

- A 'getter' method
  - is a specific type of **accessor** method and typically:
    - **contains a return statement** (as the last executable statement in the method).
    - defines a **return type**.
    - **does NOT change the object state**.

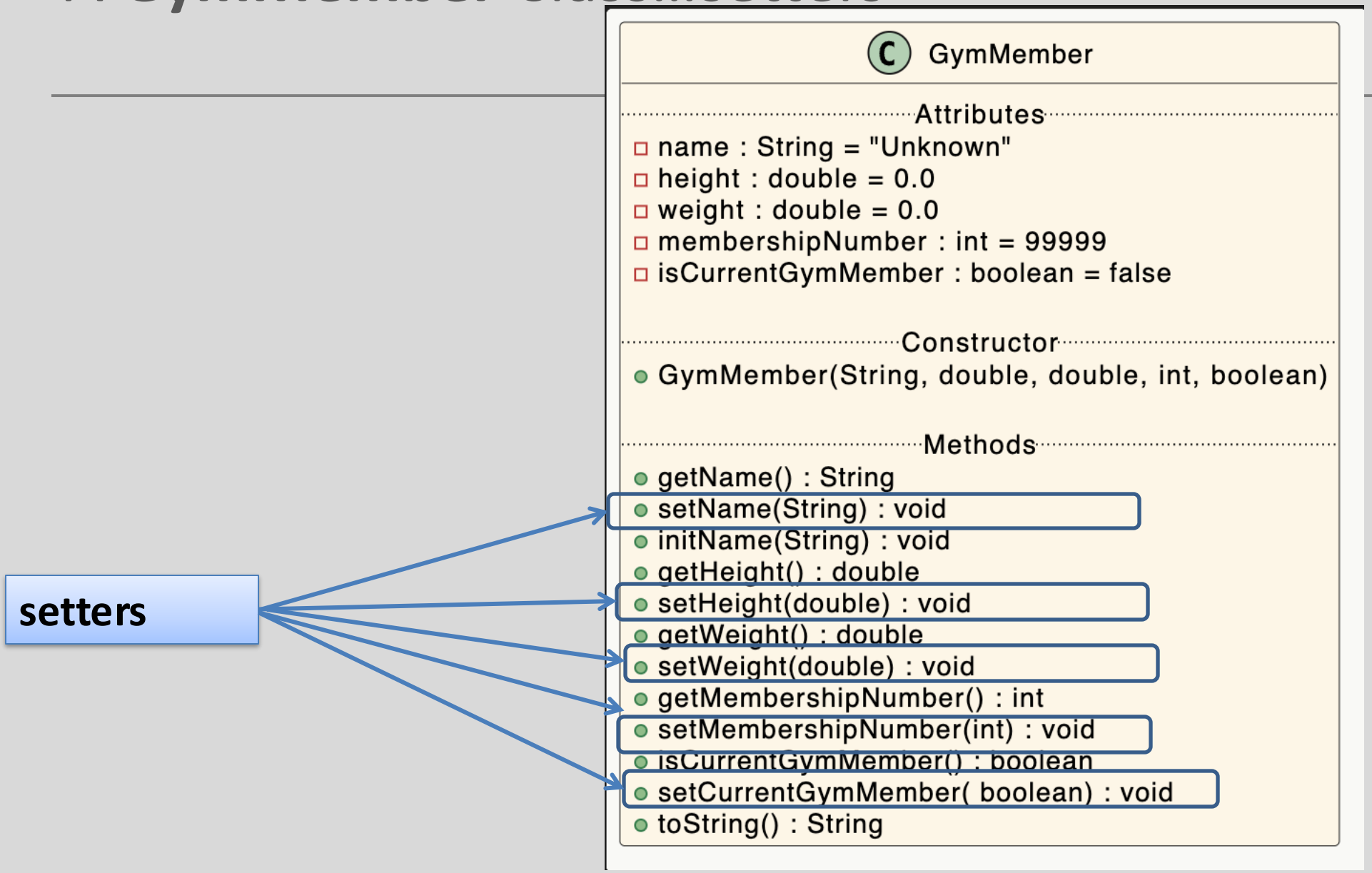
Getters


return type
visibility modifier
method name
parameter list
(empty)
public double getHeight()
{
return height;
}
return statement
start and end of method body (block)

A **GymMember** Class…getters

```
public String getName() {
    return name;
}
public double getHeight() {
    return height;
}
public double getWeight() {
    return weight;
}
public int getMembershipNumber() {
    return membershipNumber;
}
public boolean isCurrentGymMember()
{
    return isCurrentGymMember;
}
```

A **GymMember** Class…setters

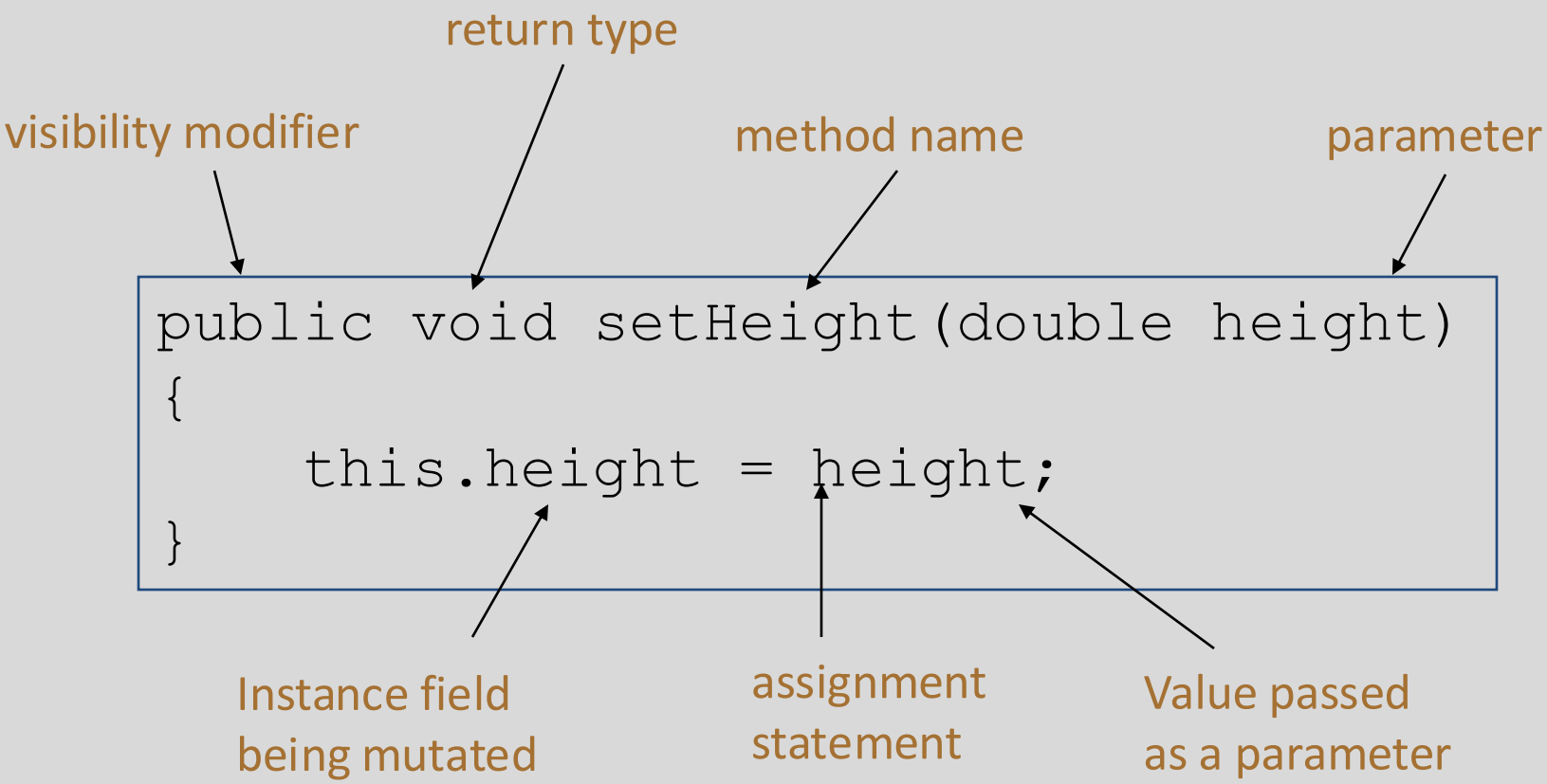


Setters (Mutator methods)

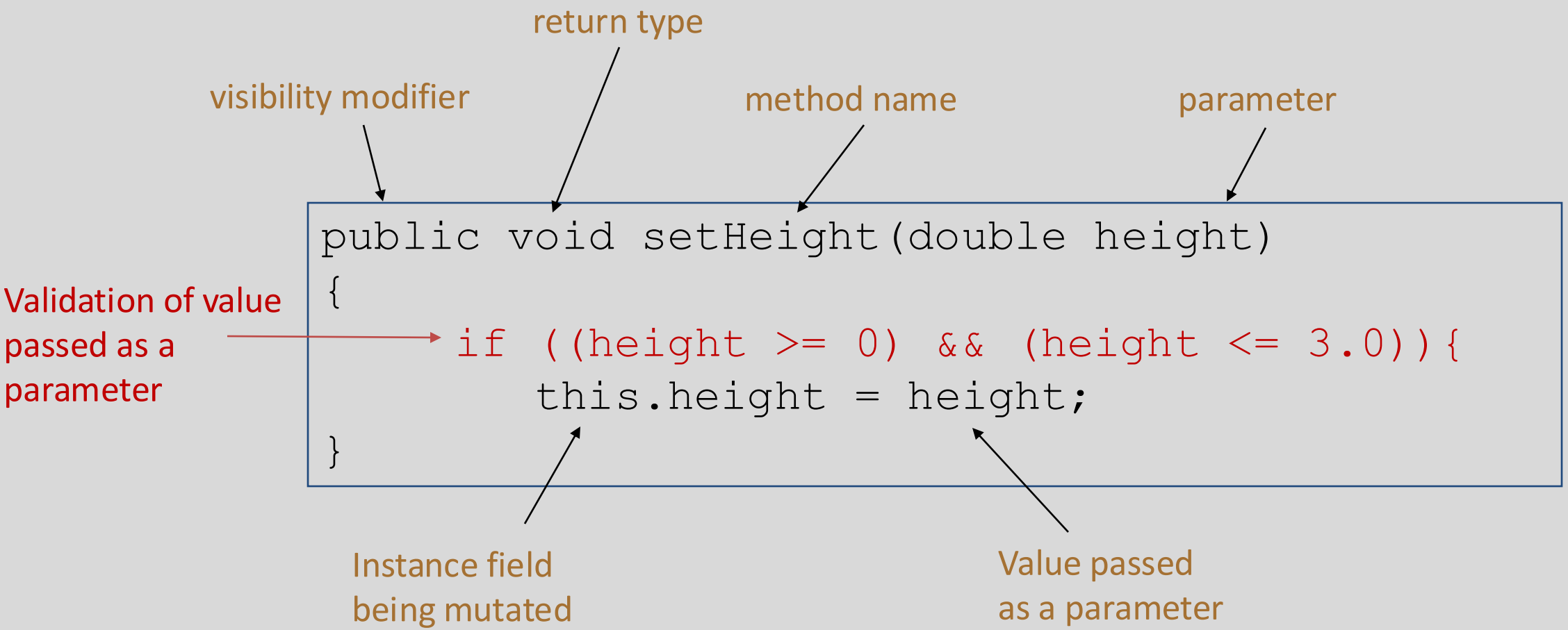
- **Mutator** methods
  - change (i.e. mutate!) an object's state.

- A 'setter' method
  - is a specific type of **mutator** method and typically:
    - contains an **assignment statement**
    - takes in a **parameter**
    - **changes the object state**.

Setters – without validation



Setters – with validation



A **GymMember** Class…special case for Strings in Constructor

```
public void initName(String name) {
    if (name != null){
        if (name.length() > 30)
            this.name = name.substring(0,30);
        else
            this.name = name;
        }
}
```

This is used in the constructor instead of the setter (setName()). This is because we assume that the name is somewhat correct (just too long)

We will always used this technique for Strings

A **GymMember** Class…setters

```
public void setName(String name) {
    if (name != null){
        if (name.length() <= 30)
            this.name = name;
}
}
```

A **GymMember** Class…setters

```
public void setHeight(double height) {
    if ((height >= 0) && (height <= 3.0)) {
        this.height = height;
    }
}
```

```
public void setWeight(double weight) {
    if ((weight >= 0) && (weight <= 500)) {
        this.weight = weight;
    }
}
```

A **GymMember** Class…setters

```
public void setMembershipNumber(int membershipNumber) {
    if ((membershipNumber > 0) && (membershipNumber < 99999)) {
        this.membershipNumber = membershipNumber;
    }
}
```

```
public void setCurrentGymMember(boolean currentGymMember) {
    isCurrentGymMember = currentGymMember;
}
```

Getters/Setters

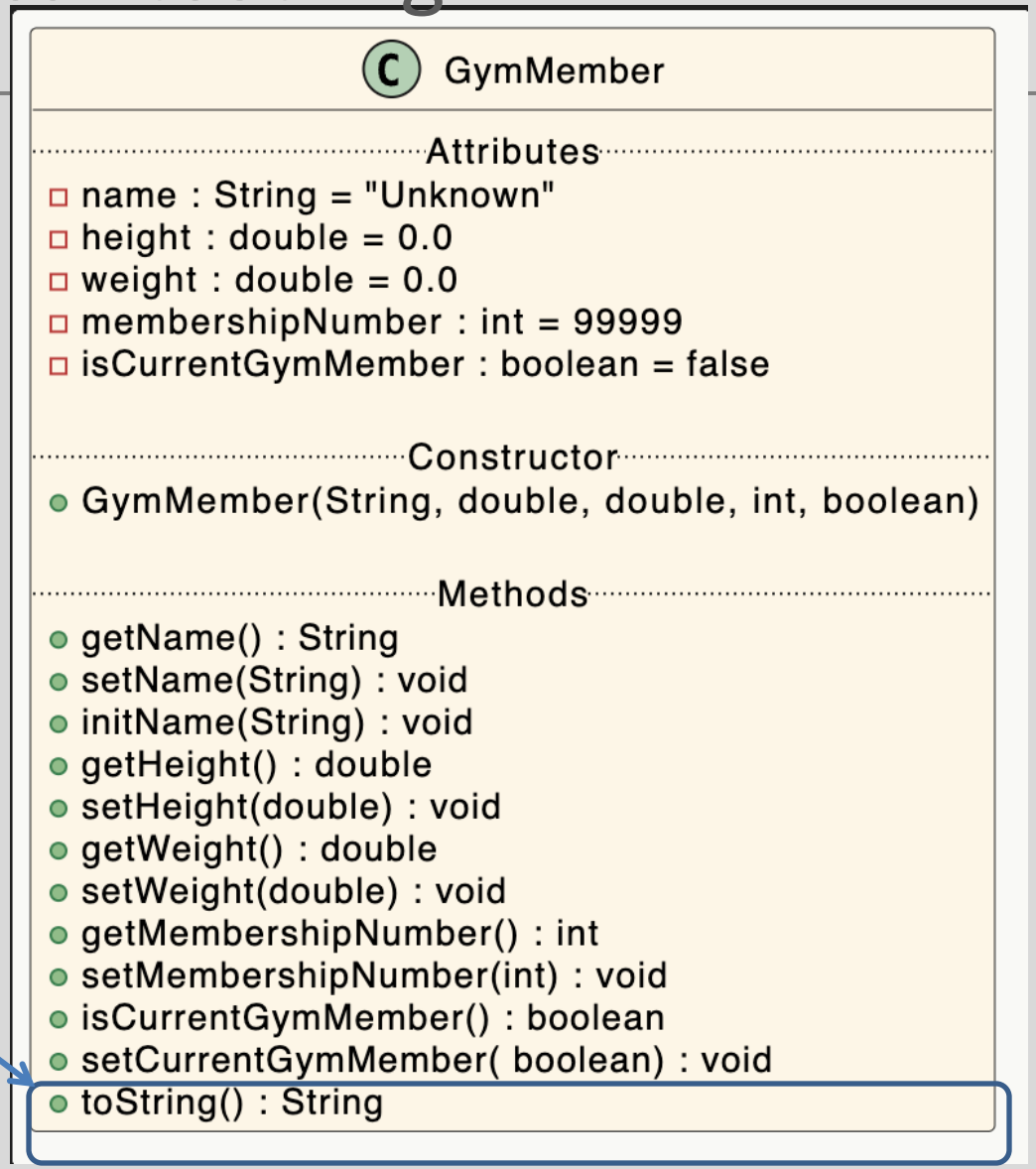
For each instance field in a class, you are normally asked to write:

- A **get**ter
  - Return statement
- A **set**ter
  - Assignment statement

A **GymMember** Class…toString

toString():

Builds and returns a String containing a user-friendly representation of the object state.



A **GymMember** Class…toString

```
@Override
public String toString() {
    return  name + ": " +
            height + "M, " +
            weight + "KG (Member Num: " +
            membershipNumber + ", current member: " +
            isCurrentGymMember + ")";
}
```

Sample Console Output :

```
Dan Sheehan: 1.91M, 111.0KG (Member Num: 222, current member: true)
```

Sample Console Output if we don't have a toString written for GymMember:

```
GymMember@77459877
```

toString()

- This is a useful method and you will write a **toString**() method for most of your classes.
- **When you print an object, Java automatically calls the toString() method** e.g.

```
GymMember gymMember = new GymMember();

//both of these lines of code do the same thing
System.out.println(gymMember);
System.out.println(gymMember.toString());
```

ENCAPSULATION RECAP

GymMember

Encapsulation in Java – **steps 1-3**

Encapsulation Step	Approach in Java

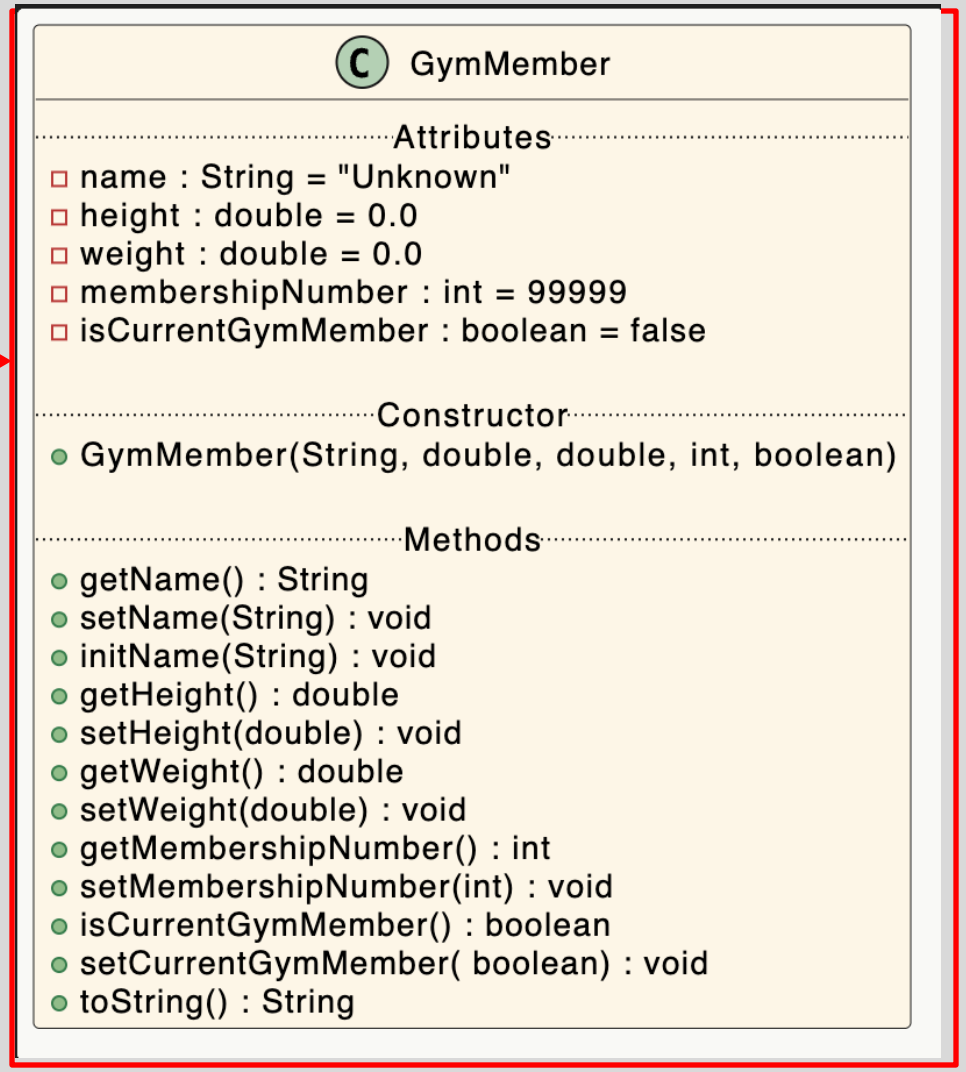
1. **Wrap** the data (fields) and code acting on the data (methods) together as single unit.	`public class ClassName`
	`{`
	`  Fields`
	`  Constructors`
	`  Methods`
	`}`

2. **Hide** the fields from other classes.	**Declare the fields of a class as private.**
3. **Access** the fields only through the methods of their current class.	**Provide public setter and getter methods** to modify and view the fields values.

http://www.tutorialspoint.com/java/java_encapsulation.htm

A GymMember Class… **An Encapsulated Class**

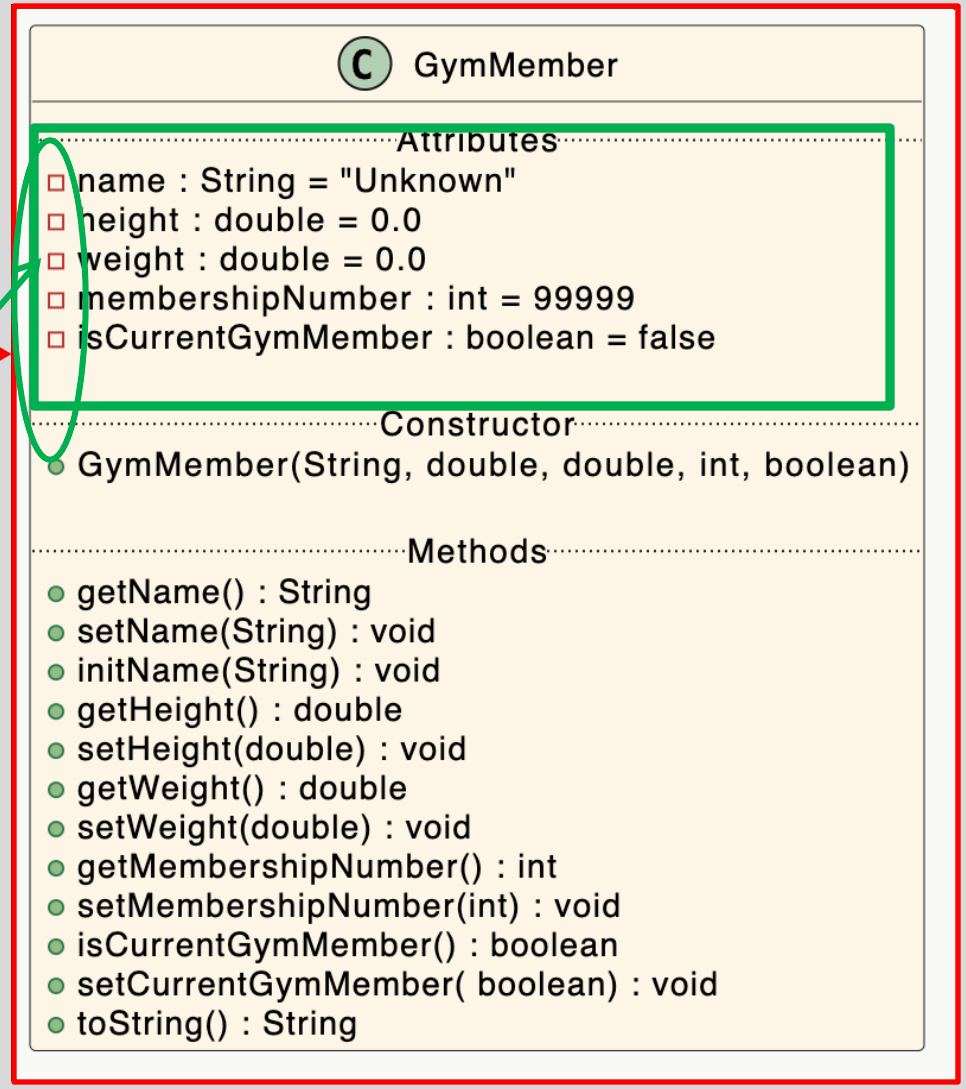
1. GymMember class **wraps** the data (fields) and code acting on the data (methods) together as **single unit**.



A GymMember Class… **An Encapsulated Class**

1. GymMember class **wraps** the data (fields) and code acting on the data (methods) together as **single unit**.

2. Fields are **hidden** from other classes.

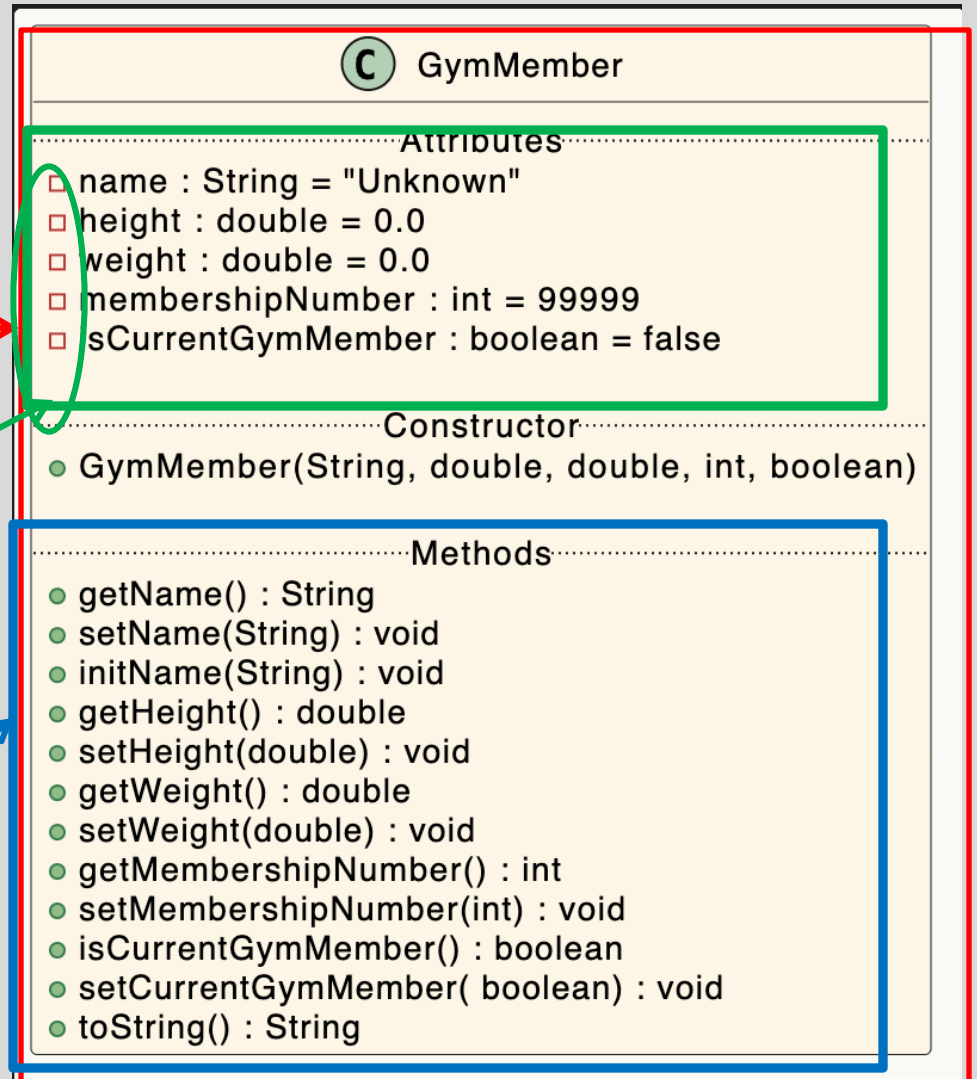


A GymMember Class… **An Encapsulated Class**

1. GymMember class **wraps** the data (fields) and code acting on the data (methods) together as **single unit**.

2. Fields are **hidden** from other classes.

3. **Access** the fields only through the methods of Product (e.g. **get**ter and **set**ter methods).



USER I/O

Driver

Using the GymMember Class

1

```
private GymMember gymMember;
```

Declaring an object **gymMember**, of type **GymMember**.

gymMember

null

Using the GymMember Class

1

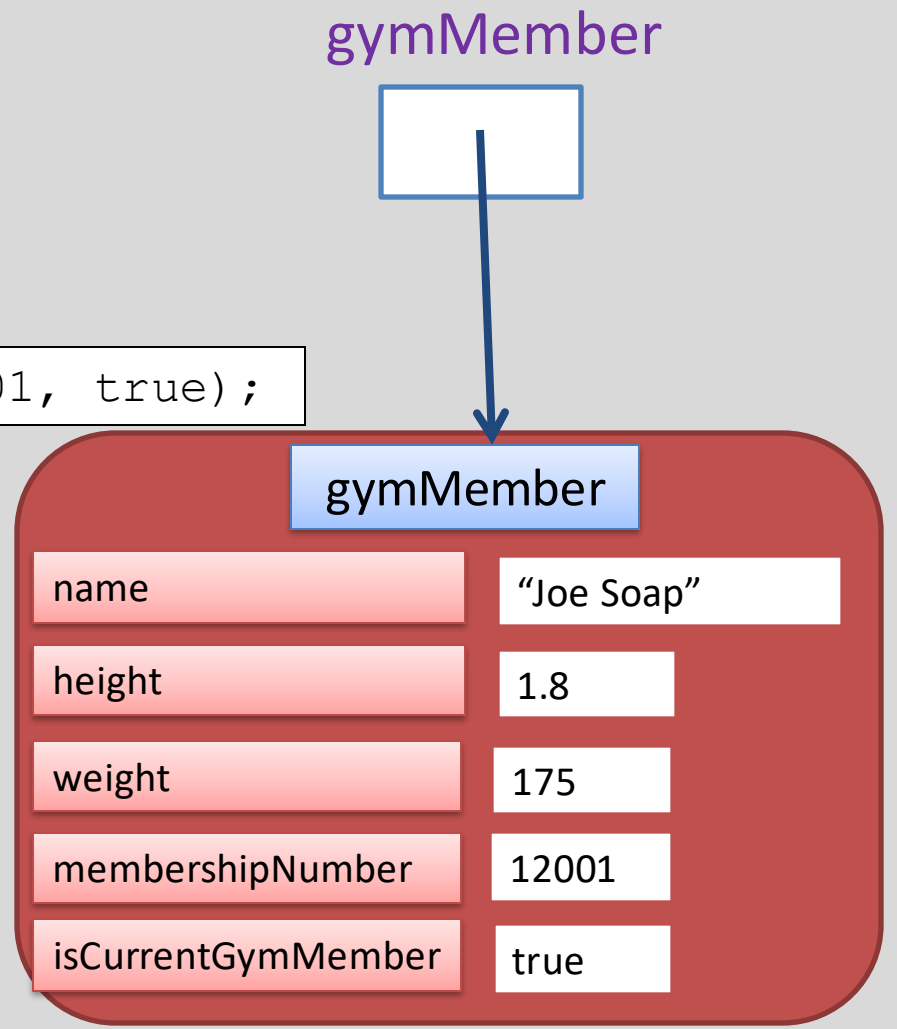
```
private GymMember gymMember;
```

Declaring an object **gymMember**, of type **GymMember**.

2

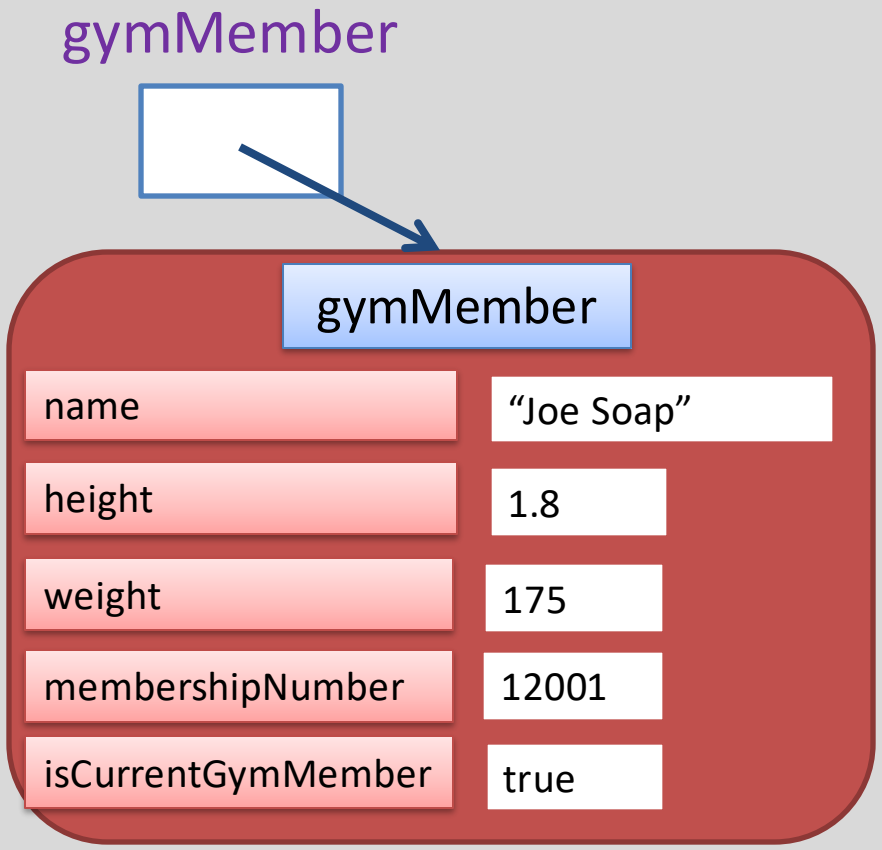
```
gymMember = new GymMember("Joe Soap", 1.8, 175, 12001, true);
```

Calls the **GymMember** *constructor* to build the **gymMember** object in memory.



Multiple GymMember objects

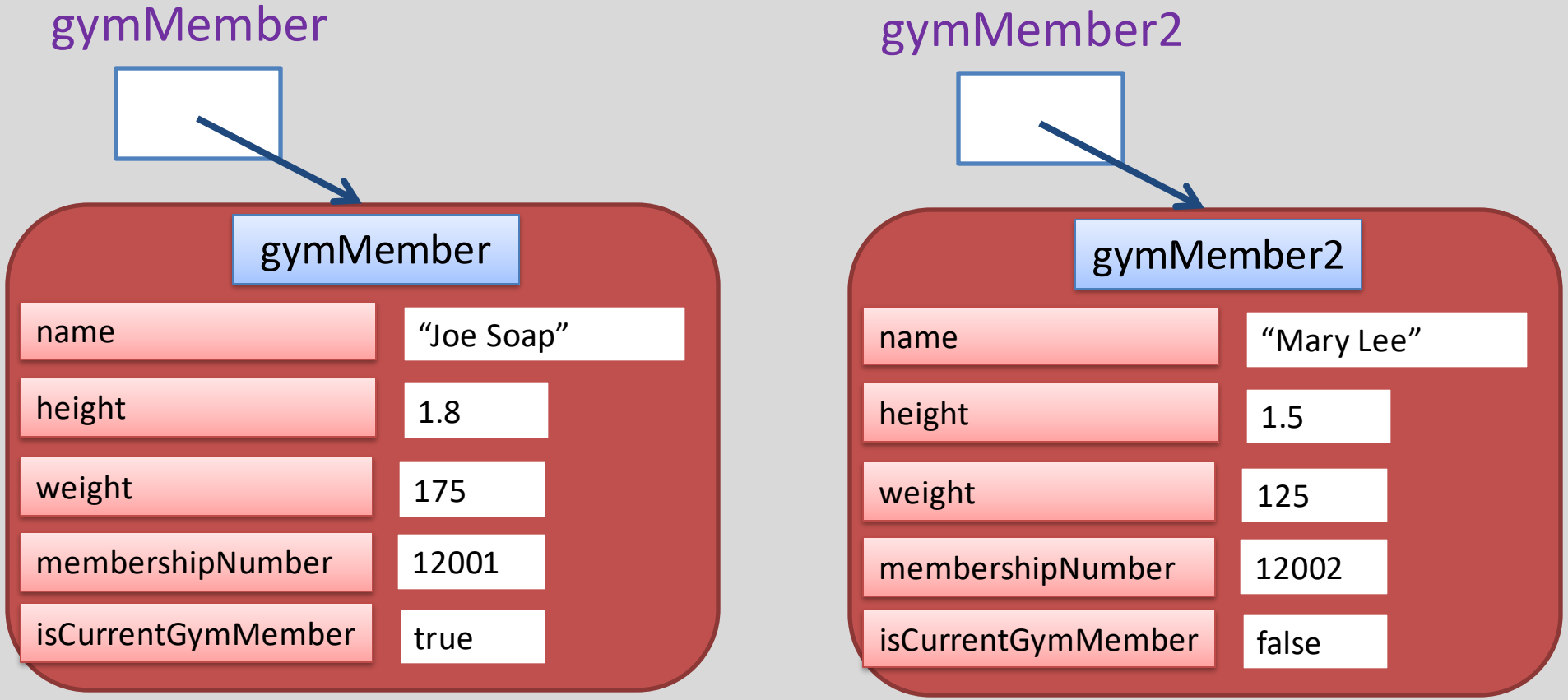
```
private GymMember gymMember = new GymMember("Joe Soap", 1.8, 175, 12001, true);
```



Multiple GymMember objects

```
private GymMember gymMember = new GymMember("Joe Soap", 1.8, 175, 12001, true);
```

```
private GymMember gymMember2 = new GymMember("Mary Lee", 1.5, 125, 12002, false);
```



Driver Class

```
import java.util.Scanner;

public class Driver {

    private Scanner input = new Scanner(System.in);
    private GymMember gymMember;

    public static void main(String[] arg) {
        new Driver();
    }

    public Driver() {
        addGymMember();
        printGymMember();
    }

    // addGymMember () code
    // printGymMember() code

}
```

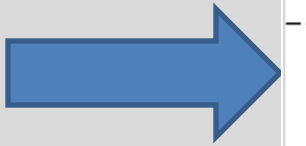
Driver Class

```
private void addGymMember() {
   //obtaining the data from the user
   System.out.println("Entering details");
   System.out.println("----------------");
   System.out.print("   Enter your name:              ");
   String name = input.nextLine();
   System.out.print("   Enter your height (meters):  ");
   double height = input.nextDouble();
   System.out.print("   Enter your weight (kgs):     ");
   double weight = input.nextDouble();
   System.out.print("   Enter the membership number: ");
   int membershipNumber = input.nextInt();
   System.out.print("   Is current member (y/n):     ");
   char isCurrentMemberChar = input.next().charAt(0);

   boolean isCurrentMember = false;
   if ((isCurrentMemberChar == 'Y') || (isCurrentMemberChar == 'y')) {
      isCurrentMember = true;
   }

   gymMember = new GymMember(name, height, weight, membershipNumber, isCurrentMember);
}
```

```
Entering details
----------------
   Enter your name:              Joe Soap
   Enter your height (meters):   1.7
   Enter your weight (kgs):      75
   Enter the membership number:  10001
   Is current member (y/n):      y
```

```java
private void printGymMember()
{
   //printing out the data to the user
   System.out.println("\n\nPrinting details");
   System.out.println("----------------");
   System.out.println(gymMember);
}
```

Driver Class



```
Driver.main({ });
Entering details
----------------
   Enter your name:              Joe Soap
   Enter your height (meters):   1.7
   Enter your weight (kgs):      75
   Enter the membership number:  10001
   Is current member (y/n):      y


Printing details
----------------
Joe Soap: 1.7M, 75.0KG (Member Num: 10001, current member: true)
```

Questions?