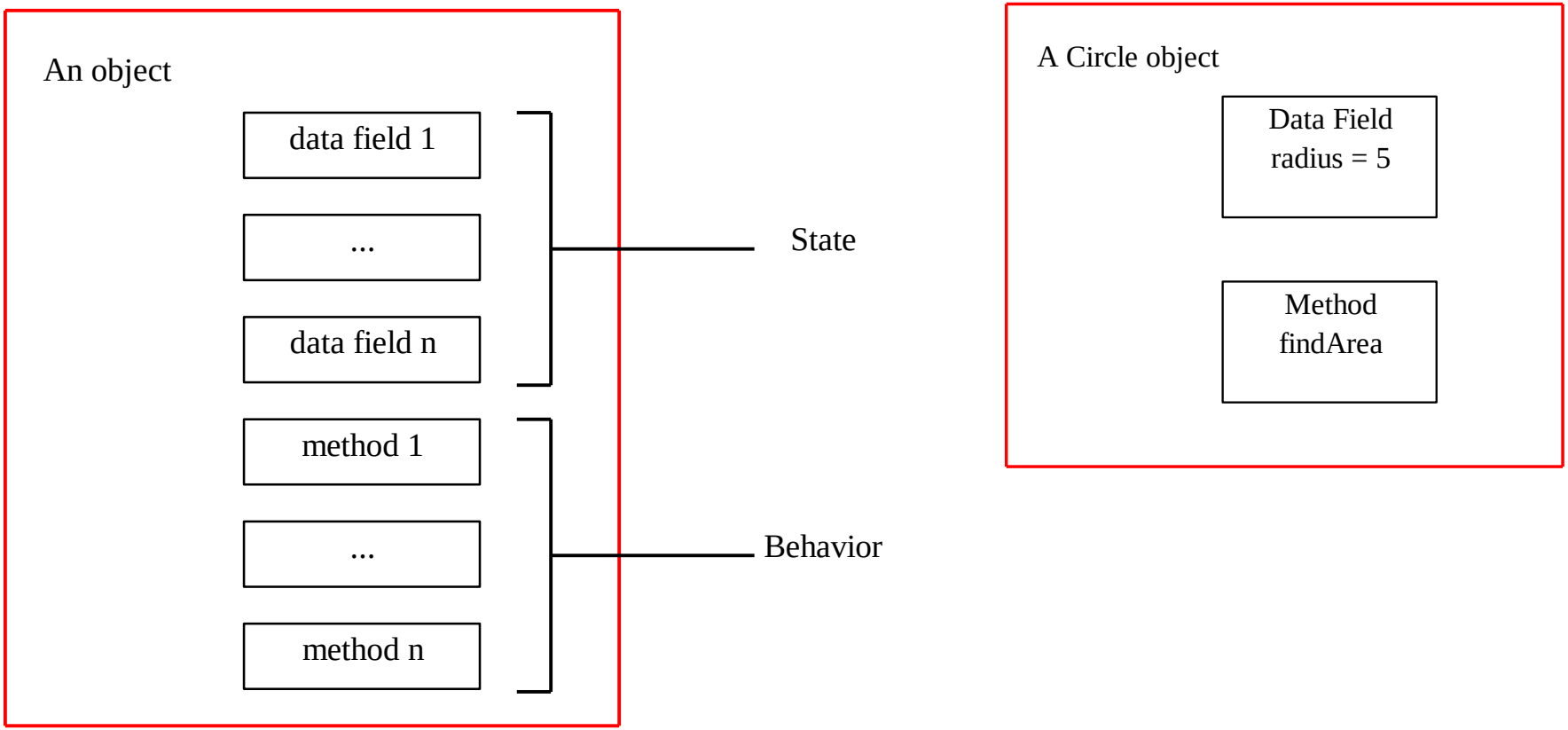
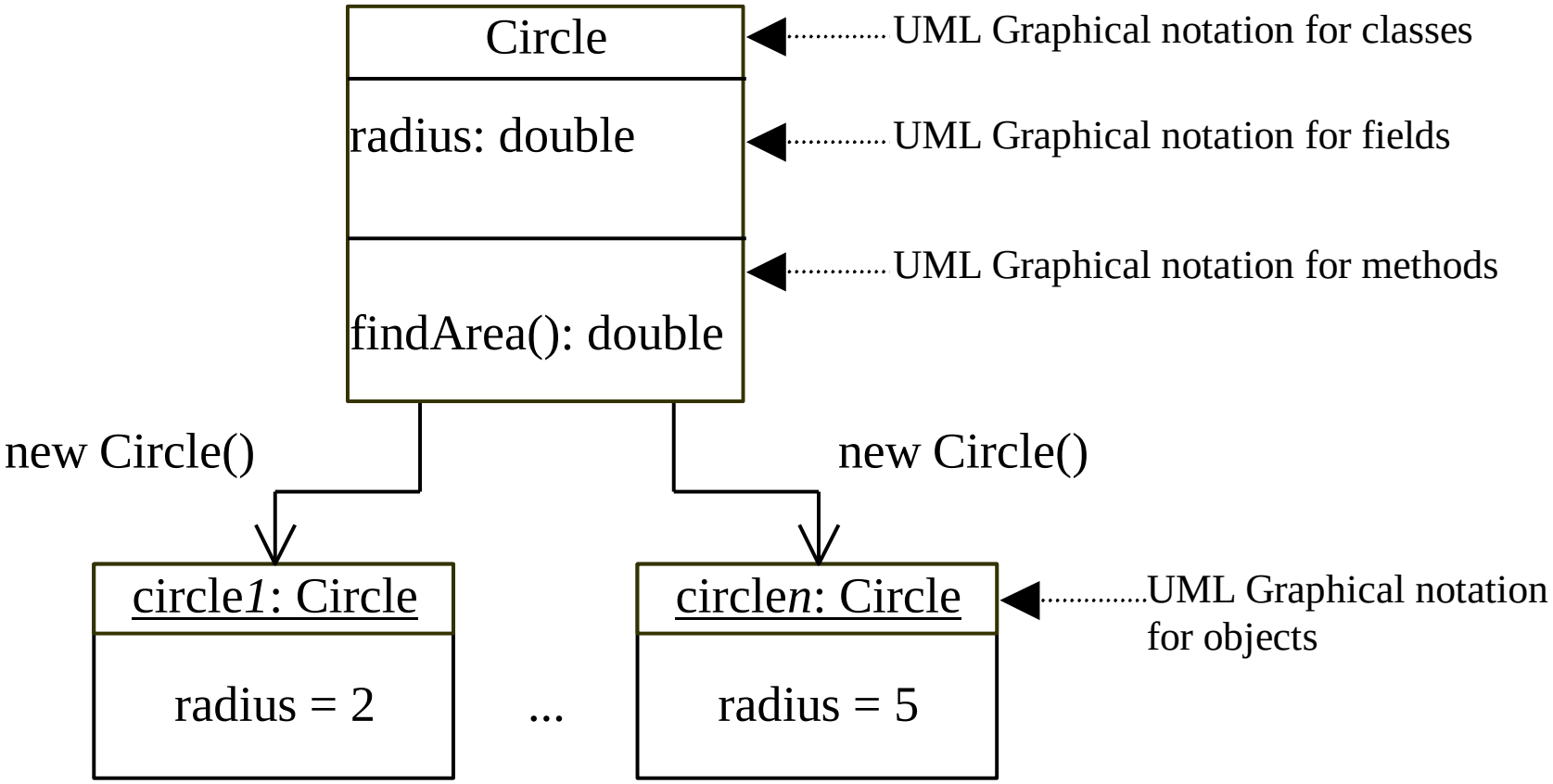
Objects and Classes

- OO Programming Concepts
- Creating Objects and Object Reference Variables
  - Differences between primitive data type and object type
  - Automatic garbage collection
- Constructors
- Modifiers (`public`, `private` and `static`)
- Instance and Class Variables and Methods
- Scope of Variables
- Use the this Keyword

OO Programming Concepts



Class and Objects



Class Declaration

```
class Circle {
   double radius = 1.0;

   double findArea(){
      return radius * radius * 3.14159;
   }
}
```

Declaring Object Reference Variables

```
ClassName objectReference;
```

Example:

```
Circle myCircle;
```

Creating Objects

```
objectReference = new ClassName();
```

Example:

```
myCircle = new Circle();
```

The object reference is assigned to the object reference variable.

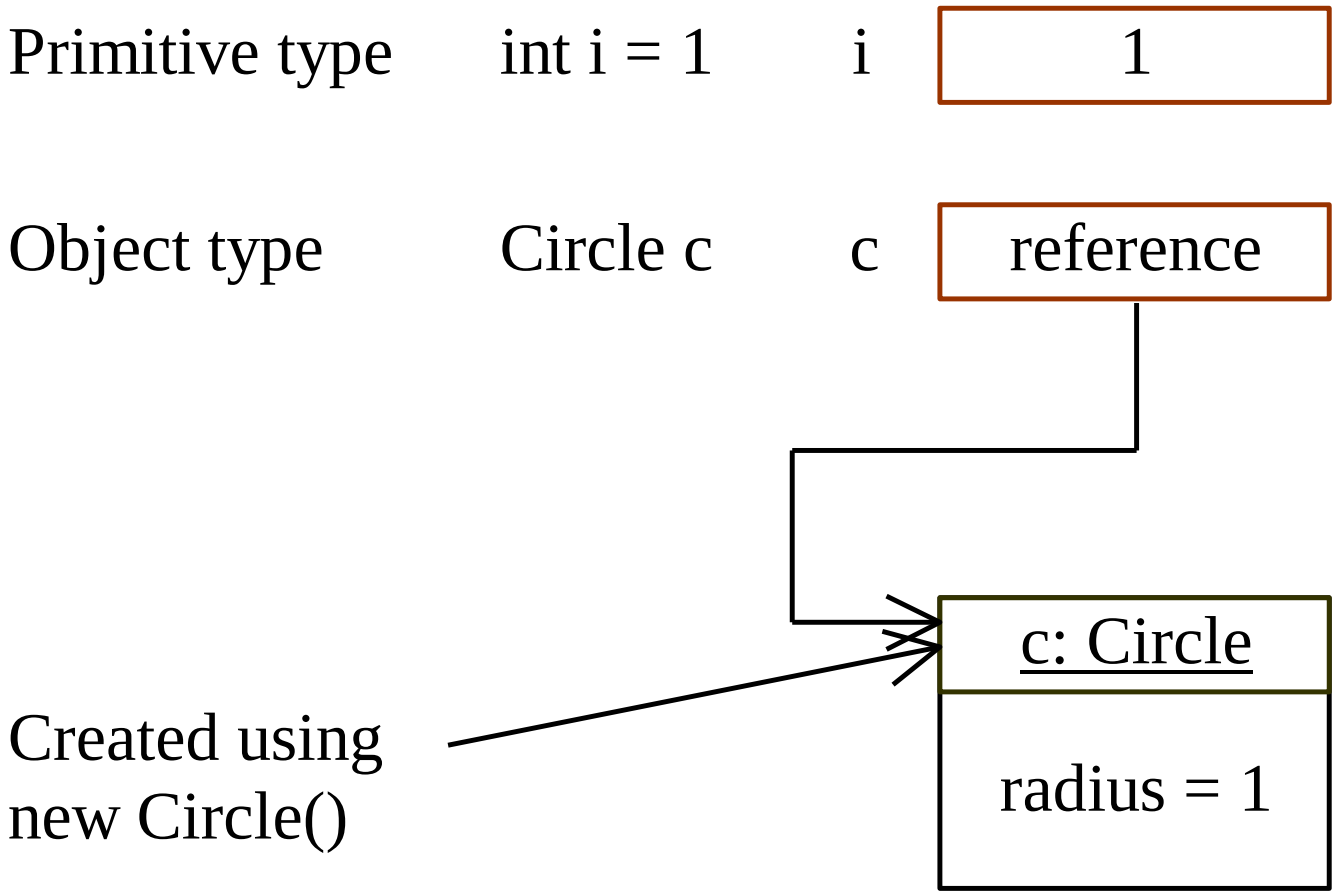
Declaring/Creating Objects in a Single Step

```
ClassName objectReference = new ClassName();
```

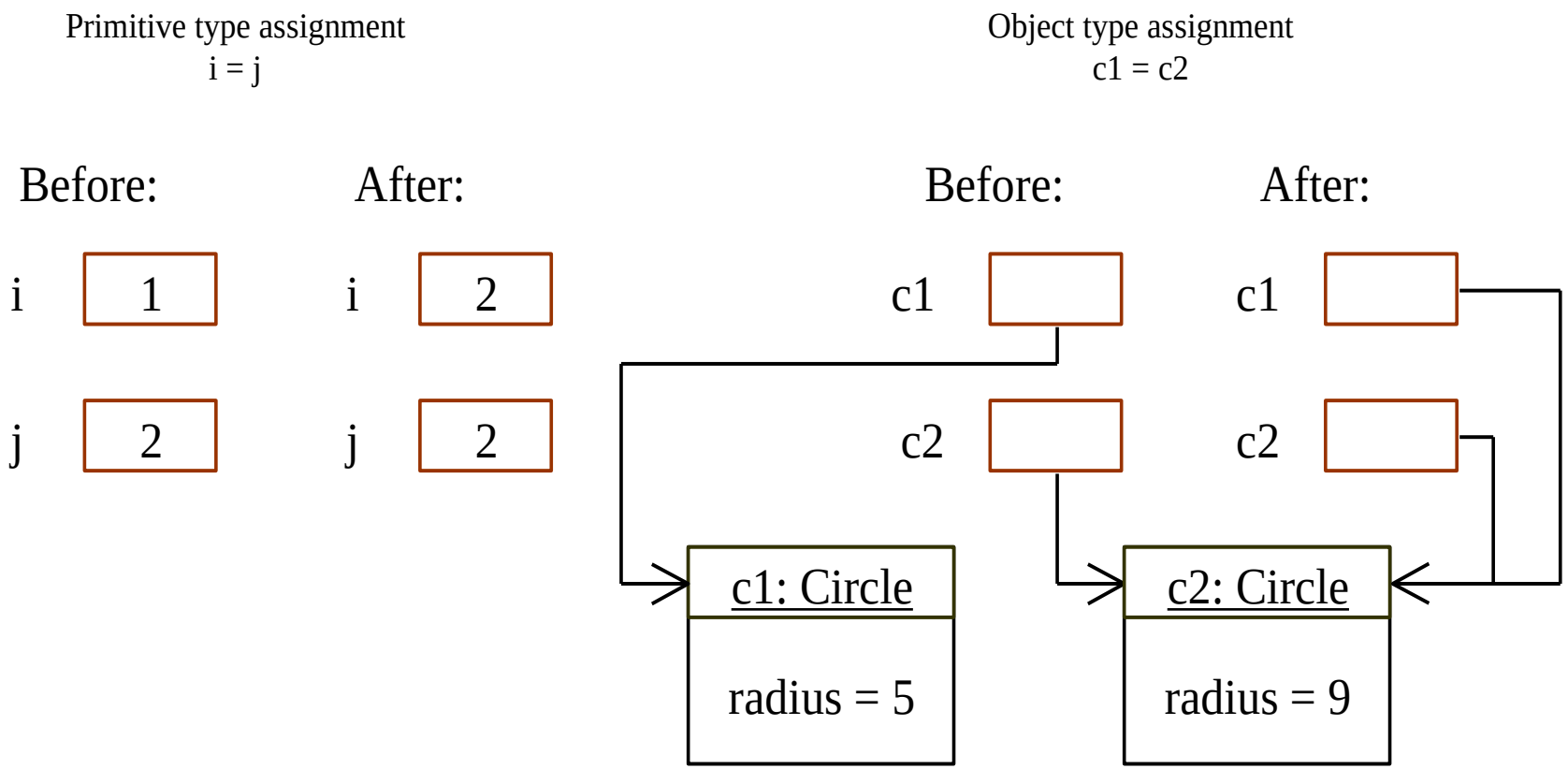
Example:

```
Circle myCircle = new Circle();
```

Differences between variables of primitive Data types and object types



Copying Variables of Primitive Data Types and Object Types



Garbage Collection

As shown in the previous figure, after the assignment statement c1 = c2, c1 points to the same object referenced by c2. The object previously referenced by c1 is no longer useful. This object is known as garbage. Garbage is automatically collected by JVM.

Garbage Collection, cont

If you know that an object is no longer needed, you can explicitly assign null to a reference variable for the object. The Java VM will automatically collect the space if the object is not referenced by any variable.

Accessing Objects

- Referencing the object's data:

```
objectReference.data
myCircle.radius
```

- Invoking the object's method:

```
objectReference.method
myCircle.findArea()
```

Constructors

```
Circle(double r) {
  radius = r;
}

Circle() {
  radius = 1.0;
}

myCircle = new Circle(5.0);
```

Constructors are a special kind of methods that are invoked to construct objects.

Constructors, cont.

A constructor with no parameters is referred to as a *default constructor*.

- Constructors must have the same name as the class itself.
- Constructors do not have a return type—not even void.
- Constructors are invoked using the new operator when an object is created. Constructors play the role of initializing objects.

Visibility Modifiers and Accessor Methods

By default, the class, variable, or data can be accessed by any class in the same package.

- `public`

  The class, data, or method is visible to any class in any package.

- `private`

  The data or methods can be accessed only by the declaring class.

The get and set methods are used to read and modify private properties.

Instance Variables, and Methods

Instance variables belong to a specific instance.

Instance methods are invoked by an instance of the class.

Class Variables, Constants, and Methods

Class variables are shared by all the instances of the class.

Class methods are not tied to a specific object.

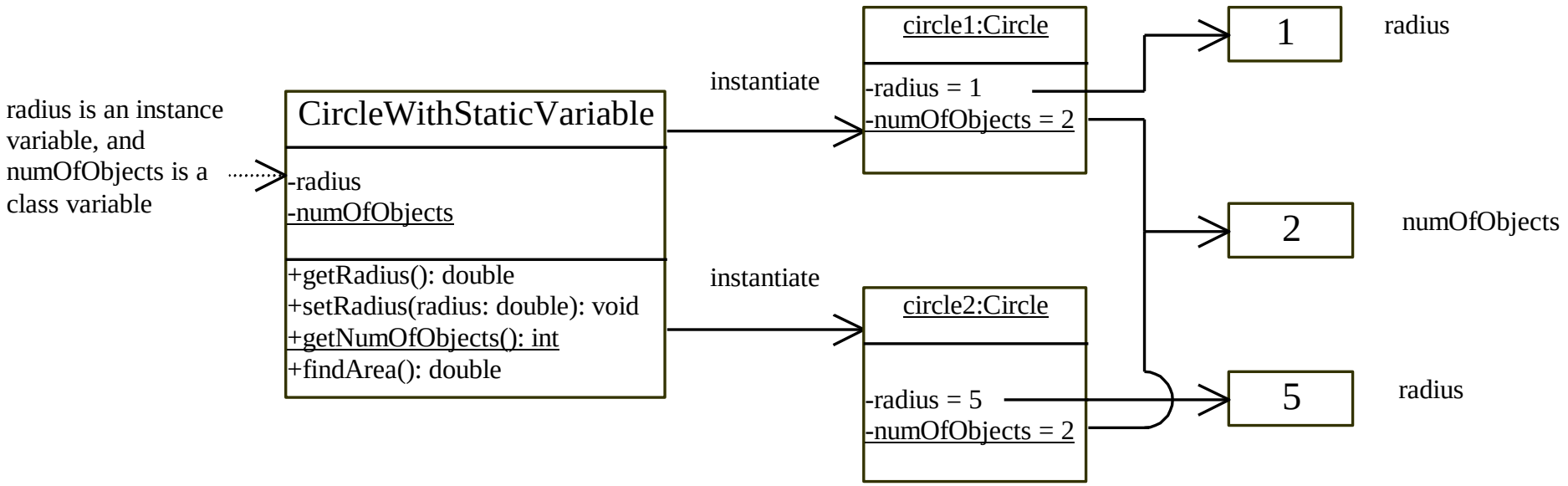
Class constants are final variables shared by all the instances of the class.

Class Variables, Constants, and Methods, cont.

To declare class variables, constants, and methods, use the static modifier.

Class Variables, Constants, and Methods, cont.

UML Notation:
+: public variables or methods
-: private variables or methods
underline: static variables or metods



Scope of Variables

- The scope of instance and class variables is the entire class. They can be declared anywhere inside a class.
- The scope of a local variable starts from its declaration and continues to the end of the block that contains the variable. A local variable must be declared before it can be used.

The Keyword this

- Use this to refer to the current object.
- Use this to invoke other constructors of the object.

Array of Objects

```
Circle[] circleArray = new Circle[10];
```

An array of objects is actually an *array of reference variables*. So invoking circleArray[1].findArea() involves two levels of referencing as shown in the next figure. circleArray references to the entire array. circleArray[1] references to a Circle object.

Array of Objects, cont.

```
Circle[] circleArray = new Circle[10];
```