JAC444 - Lecture 2

Classes

Segment 1

Objectives

Upon completion of this lecture, you should be able to:

- Define Classes and Interfaces
- Design Inner and Anonymous Classes
- Create and Work with Generic Classes

Classes – Basics

In this segment you will be learning about:

- Class Structure
- Member Variables
- Constructors
- Methods

Class Declaration

The class declaration in Java has the following format:

```
class ClassName {
    field(s)
    constructor(s)
    method declaration(s)
    other class declaration(s)
}
```

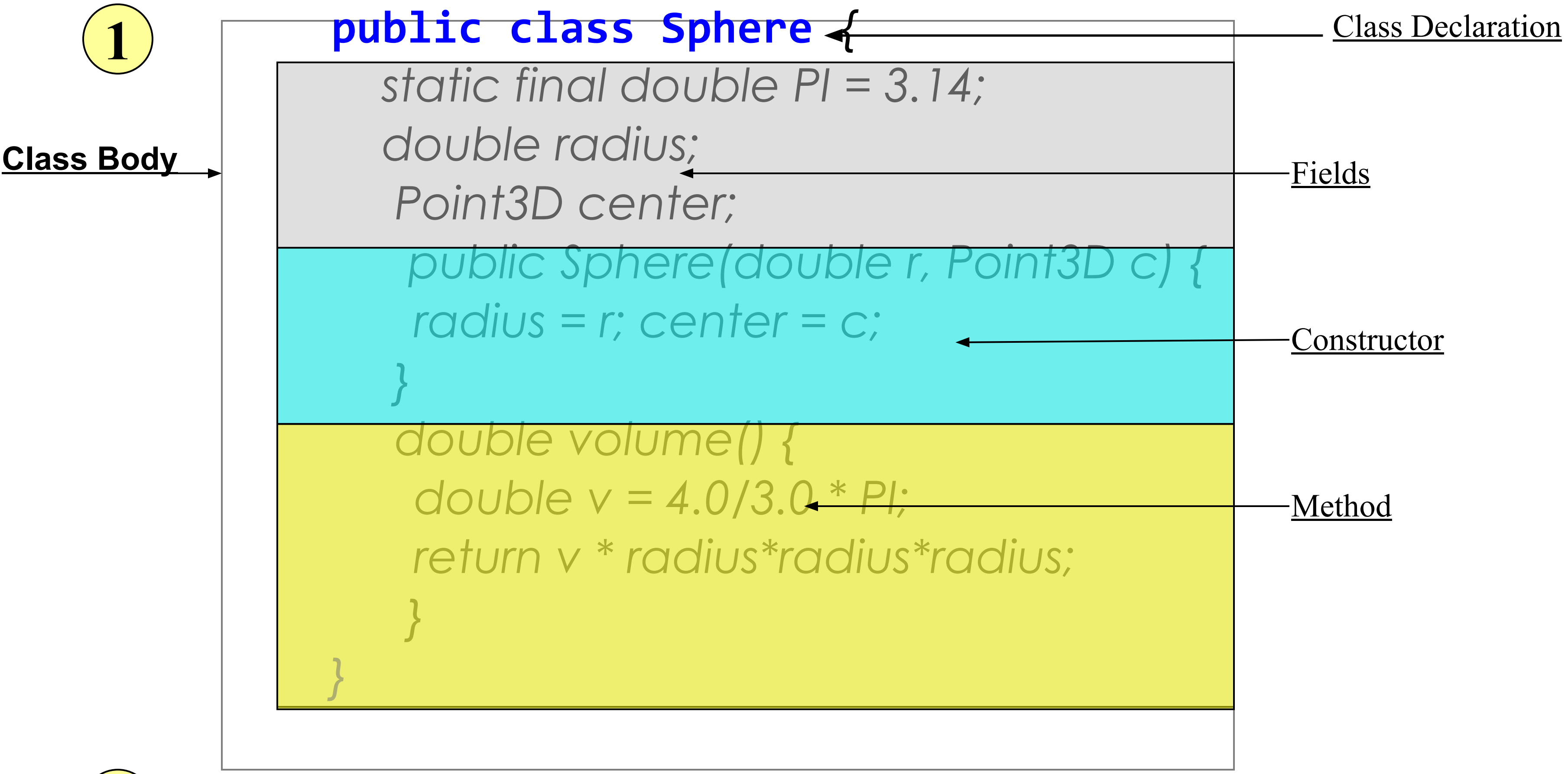
The class contains all the code you have to write. Your code must be enclosed by curly braces.

The object's' life cycle is determined by the elements of the class as following:

1. Objects initializations – Constructors
2. Objects states – Fields
3. Class and its objects behaviors – Methods

Class Structure

The fundamental unit of programming in Java is `class`.

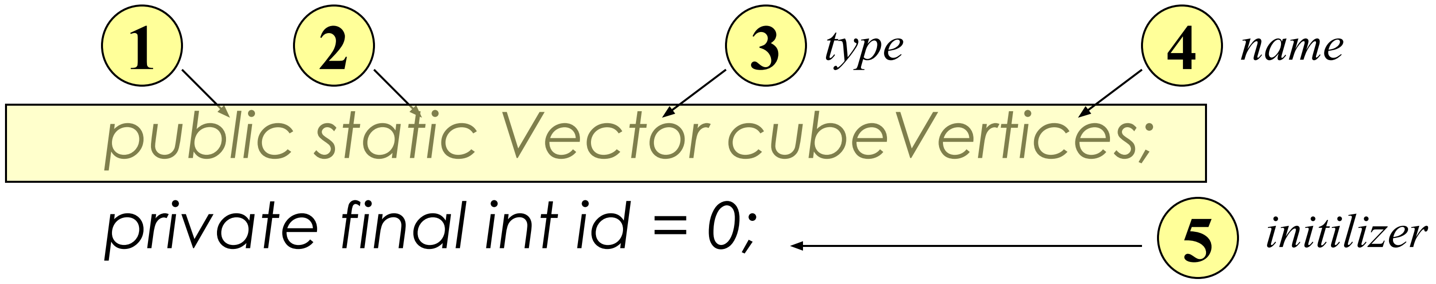


```
1   public class Sphere {
        static final double PI = 3.14;
        double radius;
         Point3D center;
          public Sphere(double r, Point3D c) {
          radius = r; center = c;
        }
        double volume() {
          double v = 4.0/3.0 * PI;
          return v * radius*radius*radius;
         }
    }
```

1 Modifiers : *public, abstract, final, strictfp*

Member Variables

Member variables declaration appears within the class body but outside of any methods or constructors.



```
public static Vector cubeVertices;
private final int id = 0;
```

1 Access modifiers: ***public, protected, private***

2 Attributes

static – variable is a class variable

final – the value of the variable cannot be changed

transient – marker used in object serialization

volatile – prevent compiler from optimization

Access Modifiers

Constructors:

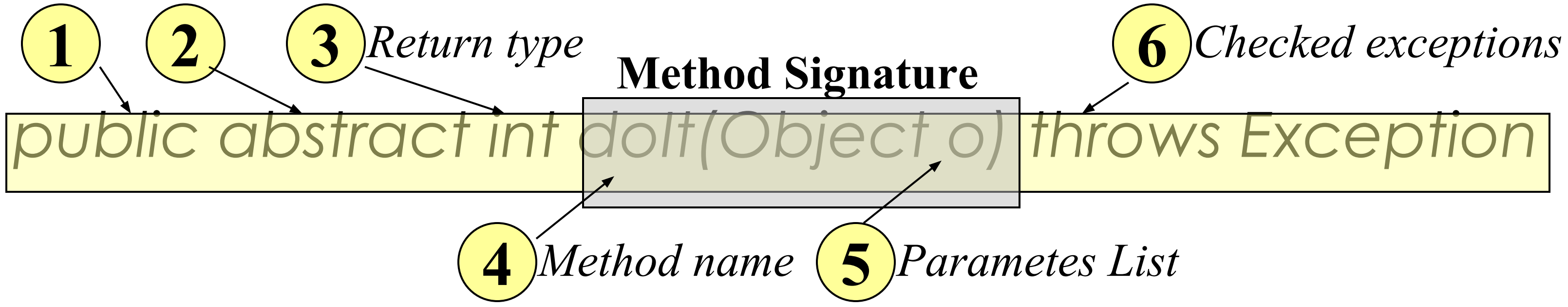
Must have the same name as the class, zero or more params. They DO NOT return anything.

`private`	No other class can instantiate the given class. The class may contain public class factory methods
`protected`	Only subclasses of the given class can create instances of it
`public`	Any class can create an instance of the given class.

(no access modifier) Only classes within the same package as the class can construct an instance of it.

Method Declaration

The only required elements of a method declaration are the method's name, its return type, and a pair of parentheses ( ).



```
public abstract int doIt(Object o) throws Exception
```

1 Access modifiers: ***public, protected, private,*** < >

2 Attributes

static – method is a class method

abstract – method has no implementation – abstract class

final – method cannot be overridden by a subclass

native – method implemented in a language other than Java

synchronized – method has a thread-safe implementation

strictfp – strict constraint for floating point arithmetic

Overloading

Defining more methods with the same name, but with different signature is called method overloading.

Java supports method name overloading so that multiple methods can share the same name.

Ex:

```
class ShapeRender {
   …
   void draw(Shape s) { …}
   void draw(Rectagle r) { …}
   void draw(Circle c, Point position) { …}
   …
}
```

Method Overloading - Static Polymorphism

Package

Definition: A package is a grouping of related types providing access protection and name space management

Create a package with a ***`package`*** statement at the top of every source file

Use ***`import`*** statement at the beginning of the file to work with package elements

Conventions:

Package names are written in all lowercase to avoid conflict with the names of classes or interfaces. The beginning of the package name must be a reversed Internet domain name

Example: ***on.senecacollege.cs…***

Access Levels

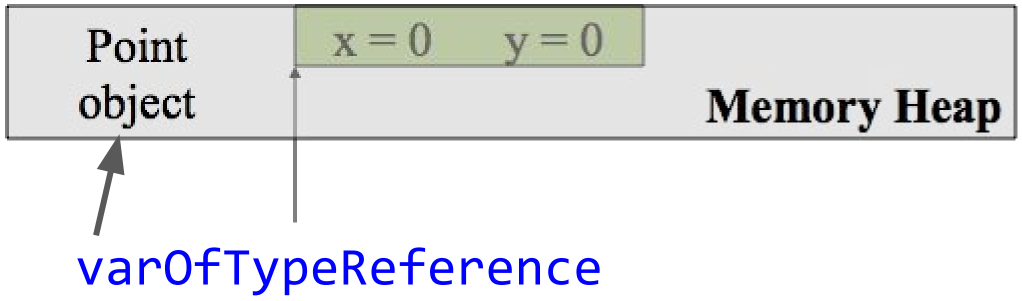
Modifier	Class	Package	Subclass	Word
public	x	x	x	x
protected	x	x	x	
no modifier	x	x		
private	x			

Creating Objects

Objects are created using the `new` keyword:

```
Point varOfTypeReference  = new Point();
```

All objects are accessed only via an object reference



Data Types in Java

There are **only two data types** in Java:

1. **Primitives**
   - boolean
   - char
   - Integer (byte, short, int, long)
   - float (float, double)

Examples:: *decimal* ***58L****, octal* ***027****, hexadecimal* ***0x8AF***

double ***18. .234E3 1.8e-1****; float* ***1.234f 0.2e-5F***

2. **References**

String ***myObject*** *= new String("Java");*

Method Invocation

A method can be invoked only through reference variable or class name (if the method is static)

```
String myObject = new String("Java");
int length = myObject.length();

Static method valueOf(...) of the String class
String val = String.valueOf(10);
```

Argument passing mechanism in java is: **pass-by-value**

Pass-by-value: a copy of each actual parameter (argument) is passed.