Java Programming

Classes & OOP

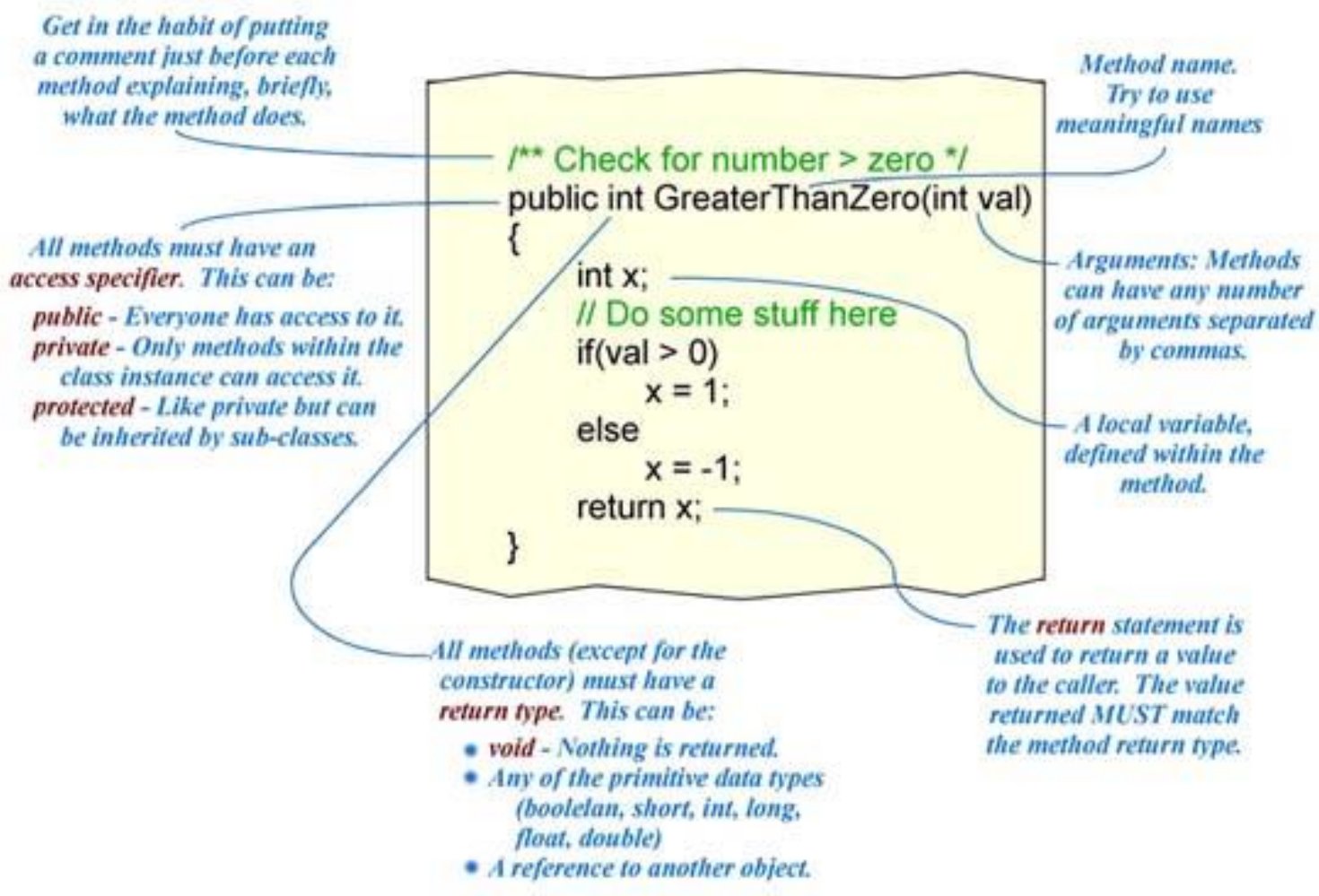
What is Object Oriented Programming (OOP)

- **Object Oriented Programming (OOP)** is a way of organizing the code in a program.
- Process of defining classes that are used to create objects which are used to perform all of the actions of the program
- Java is a programming language that is entirely object oriented.
- Other programming languages, like C++, Python, etc., do not require you to use OOP, even though most programmers do.

OOP Terms

- **Class**
  - source code file which the programmer writes that provides the "**blueprint**" for creating an object.
- **Object**
  - What is created when the program is running using the classes as "**blueprints**" or guides.
- **Instantiate**
  - The process of creating an object from a class "**blueprint**" using the **new** operator.
- **Member (instance)Variables**
  - These are variables that are created as part of a class which can be accessed and used by any method (function) in the class.
- **Member methods (functions)**
  - These are blocks of code in a class that define what a class does.
- **Constructor**
  - This is a special method included in all class files.
  - The method has the same name as the class, but has no return type.
  - When you instantiate a class in a program using the **new** operator this function gets called automatically in the new object.
  - It is here that you do all the set up and initialization needed to get this object ready to be used in your program.
- **Inheritance**
  - The ability to create a class that is a **child** class of another class. We say that the child class **extends** the parent class.
  - The child class then **inherits** from its parent all member variables and member methods from the parent class which has an access specifier of public or protected, but not those that are private.

Member Methods (functions)



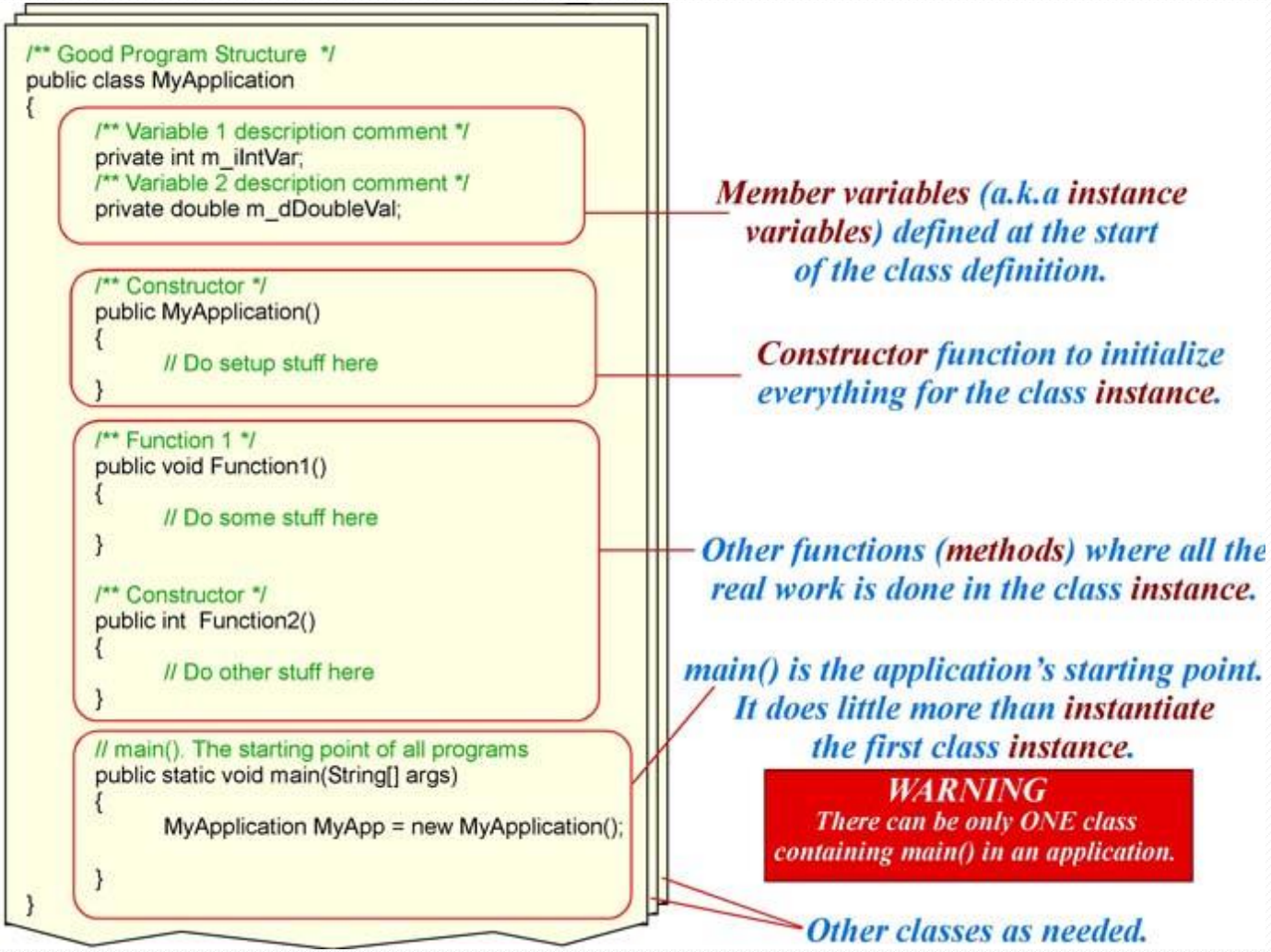
Inheritance

```
public class PitBull extends Dog
{
        //-------------------------------
        // Class Constructor
```

Class File Organization

- A class consists of the header (**public class MyApplication**) in the next slide, an opening brace, a list of member variables, a constructor method (function), any number of other methods (functions), and a closing brace.
- There must be **one and only one** class in a program that also contains a **main()** function in the format shown below.
- All **main()** should really do is instantiate one instance of the class it is in. A Java program may consist of more than one class.

Example Class File Organization



Creating Instances Using the *new* Operator

- In code, when you want to create an instance of a class, you use the **new** operator followed by what looks like a call to the class constructor of the class with appropriate arguments.
- In the image below, the **new** operator is used to create three instances of the class Dog.
- Each instance is passed three arguments: an int specifying the size of the Dog, a String giving the "type" of Dog, and a String giving the name of the Dog.

```
// Create a big dog
m_BigDog = new Dog(75, "Pit Bull", "Butch");

// Create a middle sized dog
m_MediumDog = new Dog(25, "Cocker", "Missy");

// Create a small dog
m_SmallDog = new Dog(5, "Chihuahua", "Bitsy");
```

Homework #10 Tip

Date

Calendar

GregorianCalendar

SimpleDateFormat

Example Method

```
public int getTimePeriodInYears(Date pastDate) {
    int timePeriod = 0;

    if (pastDate != null) {
        GregorianCalendar timePeriodCalendar = new GregorianCalendar();
        int pastYear = 0;
        int currentYear = 0;

        timePeriodCalendar.setTime(pastDate);
        pastYear = timePeriodCalendar.get(Calendar.YEAR);
        currentYear = new GregorianCalendar().get(Calendar.YEAR);
        timePeriod = currentYear - pastYear;
    }

    return timePeriod;
}
```

Example Invocation

```
public static void main(String args[]) {
    Date pastDate =
        new GregorianCalendar(2010, 03, 15).getTime();

    System.out.println(
        "Period in Years: "
        + getTimePeriodInYears(pastDate));
}
```

Example Formatting the Date

```
Date currDate = new Date();
SimpleDateFormat formatter =
    new SimpleDateFormat("MM/dd/yyyy");

System.out.println("Formatted Date is: "
    + formatter.format(currDate));
```

Packages to be imported…

- java.util.Date
- java.util.Calendar
- java.util.GregorianCalendar
- java.text.SimpleDateFormat