Object and Class 对象与类

Object-Oriented Programming Concept 面向对象的编程概念

An object represents an entity in the real world that can be distinctly identified from a **class/templates of objects with common properties**.

对象表示现实世界中的实体，可以从具有公共属性的对象的 **类/模板** 中明确识别。

- An object has a unique **state** and **behavior**:

  对象具有唯一的**状态**和**行为**:

  - the **state** of an object consists of **a set of data fields (properties) with their current values**

    对象的状态由一组数据字段(属性)及其当前值组成

  - the **behavior** of an object is defined by **a set of instance methods**

    对象的行为由一组实例方法定义

Classes 类

In Java **classes** are **templates** that define objects of the same type

在Java中,**类**是定义相同类型对象的**模板**

- A Java class uses:

  - **non-static/instance variables** to define data fields

    非静态/实例变量,用于定义数据字段

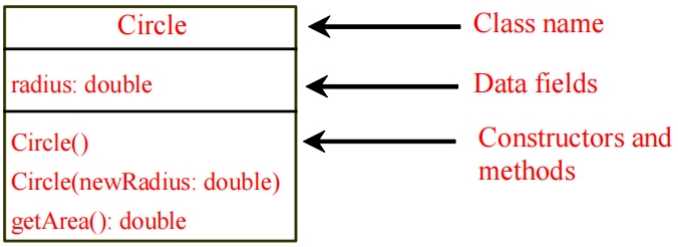
  - **non-static/instance methods** to define behaviors

    非静态/实例方法来定义行为

- A class provides a special type of methods called **constructors** which are invoked to construct objects from the class

  类提供了一种称为"构造器" 的特殊方法，调用这些方法以从类构造对象

> Object Oriented Design: The **Unified Modeling Language** (**UML**) is a general-purpose modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a object-oriented system.
>
> 面向对象设计：统一建模语言（UML）是软件工程领域的一种通用建模语言，旨在提供一种标准方法来可视化面向对象的系统的设计。



Constructor 构造器

- Constructors must have the same name as the class itself.

  构造函数必须具有与类本身相同的名称。

- Constructors do not have a return type—not even **void**.

  构造函数没有返回类型,甚至连 void 都没有。

- Constructors are invoked using the **new** operator when an object is created – they initialize objects to **reference variables**:

  创建对象时使用 new 运算符调用构造函数 – 它们将对象初始化为 引用变量：

  - **ClassName o = new ClassName();**

- Example:

  - **Circle myCircle = new Circle(5.0);**

- A class may be declared without constructors: a no-arg **default constructor** with an empty body is **implicitly** declared in the class

  可以在没有构造函数的情况下声明类：在类中通常会隐式声明具有空正文的 no-arg default constructor (无参数构造器)

静态变量和非静态变量

- Static variables and constants:

  静态变量和常量：

  ```
  static int count = 0;
  static final double PI = 3.141592;
  ```

  - global variables for the entire class: for all objects instances of this class

    整个类的全局变量：对于该类的所有对象实例

- Non-static/instance variables are date fields of objects:

  非静态/实例变量是对象的属性：

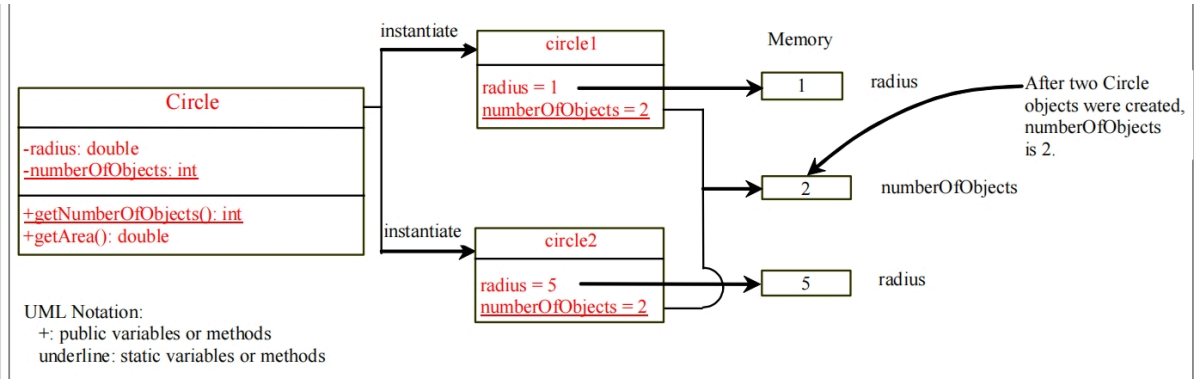
  ```
  int radius;
  double d1 = myCircle.radius;
  double d2 = yourCircle.radius;
  ```

  - Non-static/instance methods must be invoked from an object

    必须从对象调用非静态/实例方法

Static variables are shared by all the instances of the class:

静态变量由类的所有实例共享：



不同变量是否有默认值

Java assigns **no default value** to a **local variable** inside a method

Java 不为方法内的局部变量分配默认值

```
public class Test {
  public static void main(String[] args) {
    int x; // x has no default value
    String y; // y has no default value
    System.out.println("x is " + x);
    System.out.println("y is " + y);
  }
}
```

Compilation errors: the variables are not initialized

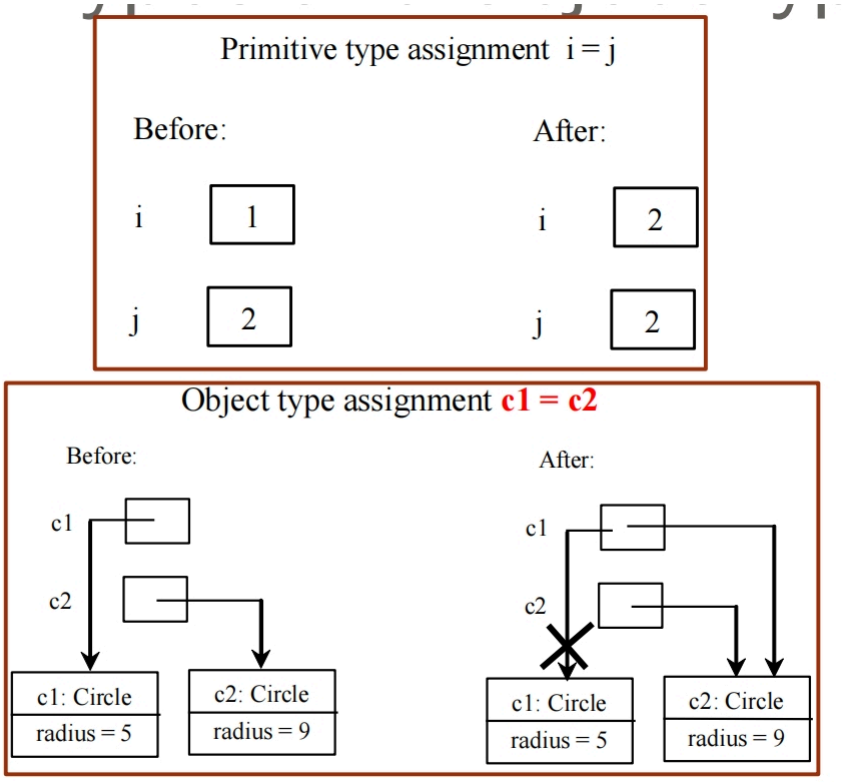
Data fields have default values

数据字段有默认值

```
public class Student {
    String name; // name has default value null
    int age; // age has default value 0
    boolean isScienceMajor; // isScienceMajor has default value false
    char gender; // c has default value '\u0000'
}
public class Test {
    public static void main(String[] args) {
        Student student = new Student();
        System.out.println("name? " + student.name); // null
        System.out.println("age? " + student.age); // 0
        System.out.println("isScienceMajor? " + student.isScienceMajor);
        // false
        System.out.println("gender? " + student.gender);
        // Note: If a data field of a reference type does not reference any
object,the data field holds a special literal value: null.
    }
}
```

Copying Variables of Primitive Data Types and Object Types

复制原始数据类型和对象类型的变量



- The object previously referenced by c1 is no longer referenced, it is called **garbage**

  之前被 c1 引用的对象不再被引用，它被称为 **垃圾**
- Garbage is automatically collected by the JVM, a process called **garbage collection**

  垃圾由 JVM 自动收集，此过程称为 **垃圾回收**
- In older languages, like C and C++, one had to explicitly deallocate/delete unused data/objects

  在较旧的语言中，如 C 和 C++，必须显式释放/删除未使用的数据/对象

Example Classes in Java

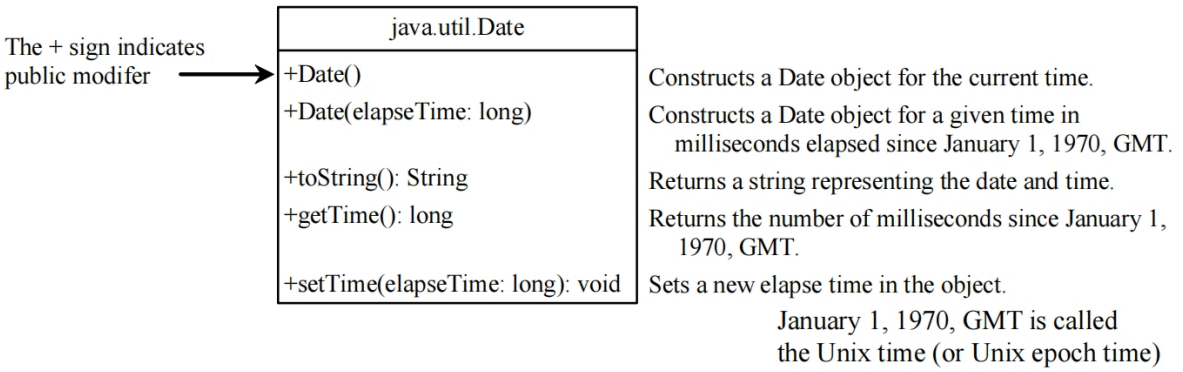
Date class 时间类

Java provides a system-independent encapsulation of date and time in the **java.util.Date** class.

Java 在 java.util.Date 类中提供了独立于系统的日期和时间封装。

The **toString** method returns the date and time as a string

toString 方法以字符串形式返回日期和时间




```
java.util.Date date = new java.util.Date();
System.out.println(date.toString());
```

Random class 随机类

java.util.Random

java.util.Random	
+Random()	Constructs a Random object with the current time as its seed.
+Random(seed: long)	Constructs a Random object with a specified seed.
+nextInt(): int	Returns a random int value.
+nextInt(n: int): int	Returns a random int value between 0 and n (exclusive).
+nextLong(): long	Returns a random long value.
+nextDouble(): double	Returns a random double value between 0.0 and 1.0 (exclusive).
+nextFloat(): float	Returns a random float value between 0.0F and 1.0F (exclusive).
+nextBoolean(): boolean	Returns a random boolean value.

```
Random random1 = new Random(3);
for (int i = 0; i < 10; i++)
    System.out.print(random1.nextInt(1000) + " ");
```

Questions

1. What is the correct code to print the value of x ?

```
public class InClassQuiz{
    int x = 5;
    static int y = 10;
    public static void main(String[] args) {
        System.out.println(???);
    }
}
```

答案应该是输入: new InClassQuiz().x。因为x是非静态变量，需要通过对象来调用

2. What is the correct code to print the value of y ?

```
public class InClassQuiz{
    int x = 5;
    static int y = 10;
    public static void main(String[] args) {
        System.out.println(???);
    }
}
```

答案是应该输入：InClassQuiz.y。因为y是静态变量，可以直接通过类名获取

3. What is the output of the following code?

```
public class Example {
    public int x;

    public static void main(String[] args) {
        Example obj1 = new Example();
        obj1.x = 5;
        Example obj2 = obj1;
        obj2.x = 10;
        System.out.println(obj1.x);
    }
}
```

应该输出: 10, 因为obj2的当前引用的对象地址和obj1相同，因此obj2做出的更改实际上obj1也会看到