第7章　7.2節
オブジェクト指向基本文法

Java 创建类

- 要创建一个类，请使用关键字class：

```
public class MyClass {
  int x = 5;
}
```

- 类应始终以大写首字母开头，并且Java文件的名称必须与public的类名一致。

Java 创建一个对象

- 我们已经创建了名为MyClass的类，现在可以使用它来创建对象。
- 要创建MyClass的对象，请指定类名称，然后指定对象名称，并使用关键字new：

```
public class MyClass {
  int x = 5;

  public static void main(String[] args) {
    MyClass myObj = new MyClass();
    System.out.println(myObj.x);
  }
}
```

多个对象

- 可以用一个类创建多个对象：

```
public class MyClass {
  int x = 5;

  public static void main(String[] args) {
    MyClass myObj1 = new MyClass();  // Object 1
    MyClass myObj2 = new MyClass();  // Object 2
    System.out.println(myObj1.x);
    System.out.println(myObj2.x);
  }
}
```

使用多个类

- 一个Java文件中可以创建多个class（推荐一个文件只有一个class），但是只能有一个public的class。
- 还可以创建一个类的对象，然后在另一个类中访问它。通常用于更好地组织类（一个类具有所有属性和方法，而另一类则具有main()方法—要执行的代码）。

```
public class MyClass {
  int x = 5;
}
class OtherClass {
  public static void main(String[] args) {
    MyClass myObj = new MyClass();
    System.out.println(myObj.x);
  }
}
```

Java 类属性（Fields）

- MyClass里的x变量实际上是类的一个属性。或称成员变量
- 一个类中可以有任意多个属性：

```
public class MyClass {
  int x = 5;
  int y = 3;
}
```

- 可以通过创建类的对象并使用点语法（.）来访问属性：

```
public class MyClass {
  int x = 5;

  public static void main(String[] args) {
    MyClass myObj = new MyClass();
    System.out.println(myObj.x);
  }
}
```

修改属性

- 还可以修改属性值：

```
public class MyClass {
  int x;

  public static void main(String[] args) {
    MyClass myObj = new MyClass();
    myObj.x = 40;
    System.out.println(myObj.x);
  }
}
```

- 如果想定义常量（不可被修改），请将该变量声明为final：

```
public class MyClass {
  final int x = 10;

  public static void main(String[] args) {
    MyClass myObj = new MyClass();
    myObj.x = 25; // will generate an error: cannot assign a value to a final
    System.out.println(myObj.x);
  }
}
```

Java 类方法[メソッド]（method）

- Java中所有的函数都是在类中声明的，并被叫做方法：

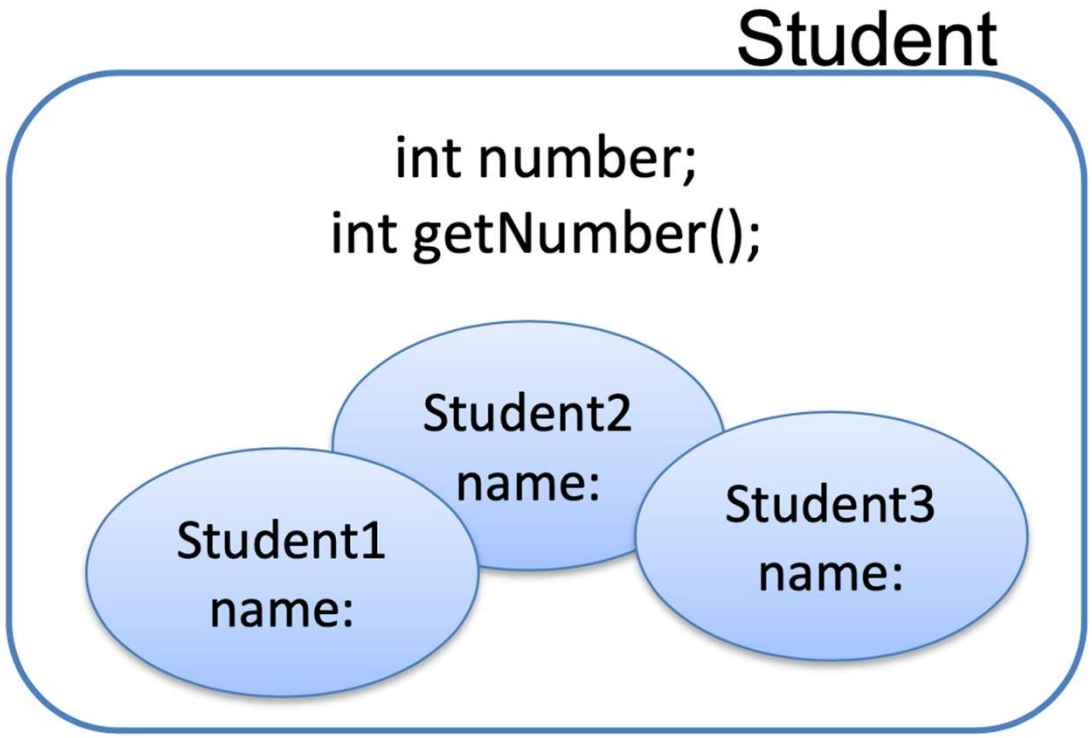
```
public class MyClass {
  static void myMethod() {
    System.out.println("Hello World!");
  }

  public static void main(String[] args) {
    myMethod();
  }
}

// Outputs "Hello World!"
```

静态方法[静的メソッド]/静态属性[静的フィールド]

- 静态的东西是属于类本身的，非静态的东西则属于实例化对象的，尝试代码：Student.java



静态与非静态的调用

- 静态属性及方法调用语法：classname.field/classname.method
- 非静态属性及方法调用语法：objname.field/objname.method

```java
public class MyClass {
  // Static method
  static void myStaticMethod() {
    System.out.println("Static methods can be called without creating objects");
  }

  // Public method
  public void myPublicMethod() {
    System.out.println("Public methods must be called by creating objects");
  }

  // Main method
  public static void main(String[] args) {
    myStaticMethod(); // Call the static method

    MyClass myObj = new MyClass(); // Create an object of MyClass
    myObj.myPublicMethod(); // Call the public method
  }
}
```

Java 构造方法[コンストラクター]（Constructor）

- Java中的构造方法[コンストラクター]是一种用于初始化对象的特殊方法。创建类的对象时将调用构造方法[コンストラクター]。它可用于设置对象属性的初始值。

```
public class MyClass {
  int x;

  // Create a class constructor for the MyClass class
  public MyClass() {
    x = 5;
  }

  public static void main(String[] args) {
    MyClass myObj = new MyClass();
    System.out.println(myObj.x);
  }
}
```

- 请注意，构造方法[コンストラクター]名称必须与类名称一致，并且不能具有返回类型（如int，void...）。
- 默认情况下，所有类都具有构造方法[コンストラクター]：如果你没有创建类构造方法[コンストラクター]，Java会自动创建默认构造方法。但默认构造方法无法设置对象属性的初始值。

构造方法[コンストラクター]参数

- 构造方法[コンストラクター]可以有任意个参数，这些参数用于初始化属性。

```
public class Car {
  int modelYear;
  String modelName;

  public Car(int year, String name) {
    modelYear = year;
    modelName = name;
  }

  public static void main(String[] args) {
    Car myCar = new Car(1969, "Mustang");
    System.out.println(myCar.modelYear + " " + myCar.modelName);
  }
}

// Outputs 1969 Mustang
```

Java 修饰符[修飾子]

- 到目前为止，出现了很多像public，static，final等关键字，他们叫做修饰符[修飾子]。

- 我们将修饰符[修飾子]分为两组：
  - 访问修饰符[アクセス修飾子] -控制访问级别
  - 非访问修饰符[アクセス以外の修飾子] -不控制访问级别，但提供其他功能

访问修饰符[アクセス修飾子]

- 有public，protected，default（默认不写），private四种
- 下面表格是可访问范围：（包即package）

	类内部	本包	子类	外部包
public	√	√	√	√
protected	√	√	√	×
default	√	√	×	×
private	√	×	×	×

- 对于类，只可以使用public或default
- 对于属性，方法和构造方法[コンストラクター]，四种都可以使用
- 尝试代码包： AccessModifiers

非访问修饰符

- 对于类，只可以用final，abstract：
  - final修饰的类：不可以被其他类继承，即不可以存在子类[サブクラス]
  - abstract修饰的类：抽象类[抽象クラス]，不可以用来生成对象的类，必须要被继承，后述。

非访问修饰符

- 对于属性和方法，可以使用final，static，abstract，transient，synchronized，volatile：
  - final：属性和方法不可以被改变/重写[オーバライド]
  - static：属性和方法属于类，不属于对象
  - abstract：只能存在于抽象类[抽象クラス]中，只能用来修饰方法。抽象方法不存在实体，比如 abstract void run(); 在抽象类[抽象クラス]的子类[サブクラス]里面提供抽象方法的实体，后述。
  - transient：序列化的对象包含被 transient 修饰的实例变量时，java 虚拟机(JVM)跳过该特定的变量。
  - synchronized：同一时间只能被一个线程访问。
  - volatile：在每次被线程访问时，都强制从共享内存中重新读取该成员变量的值。而且，当成员变量发生变化时，会强制线程将变化值回写到共享内存。这样在任何时刻，两个不同的线程总是看到某个成员变量的同一个值。

尝试代码包： Non-AccessModifiers