Introduction to Java Programming Language

Ningzhi Zhu
ShanghaiTech University

2018-09-23

Agenda

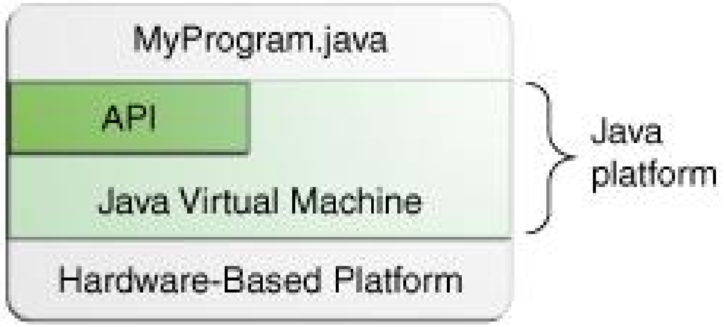
Introduction

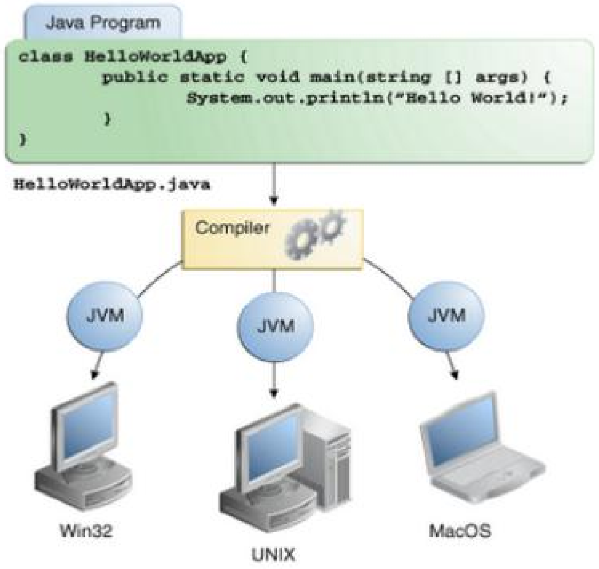
- Characteristics of Java
  - “Write Once, Run Anywhere”
  - Simple
  - Object oriented
  - Multithreaded
  - Secure
  - Dynamic

Agenda

Java platform

- “Write Once, Run Anywhere”



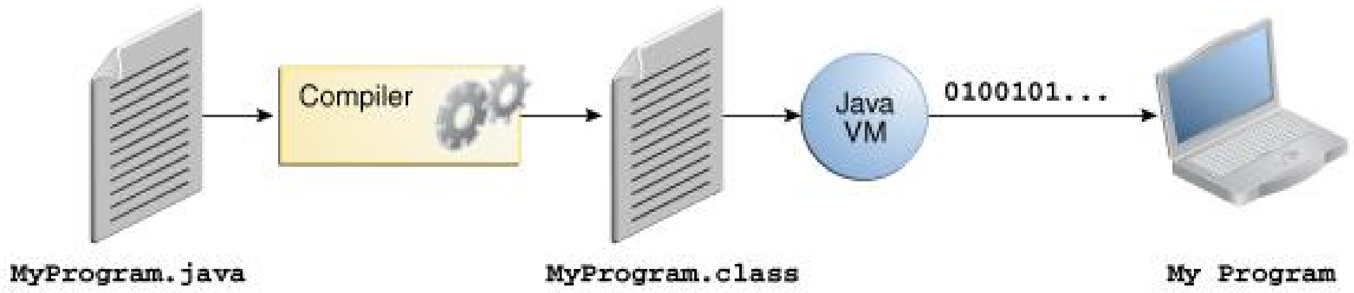


```
class HelloWorldApp {
        public static void main(string [] args) {
                System.out.println(“Hello World!”);
        }
}
```

Cited from “junji zhi university of toronto”

Java Development Process

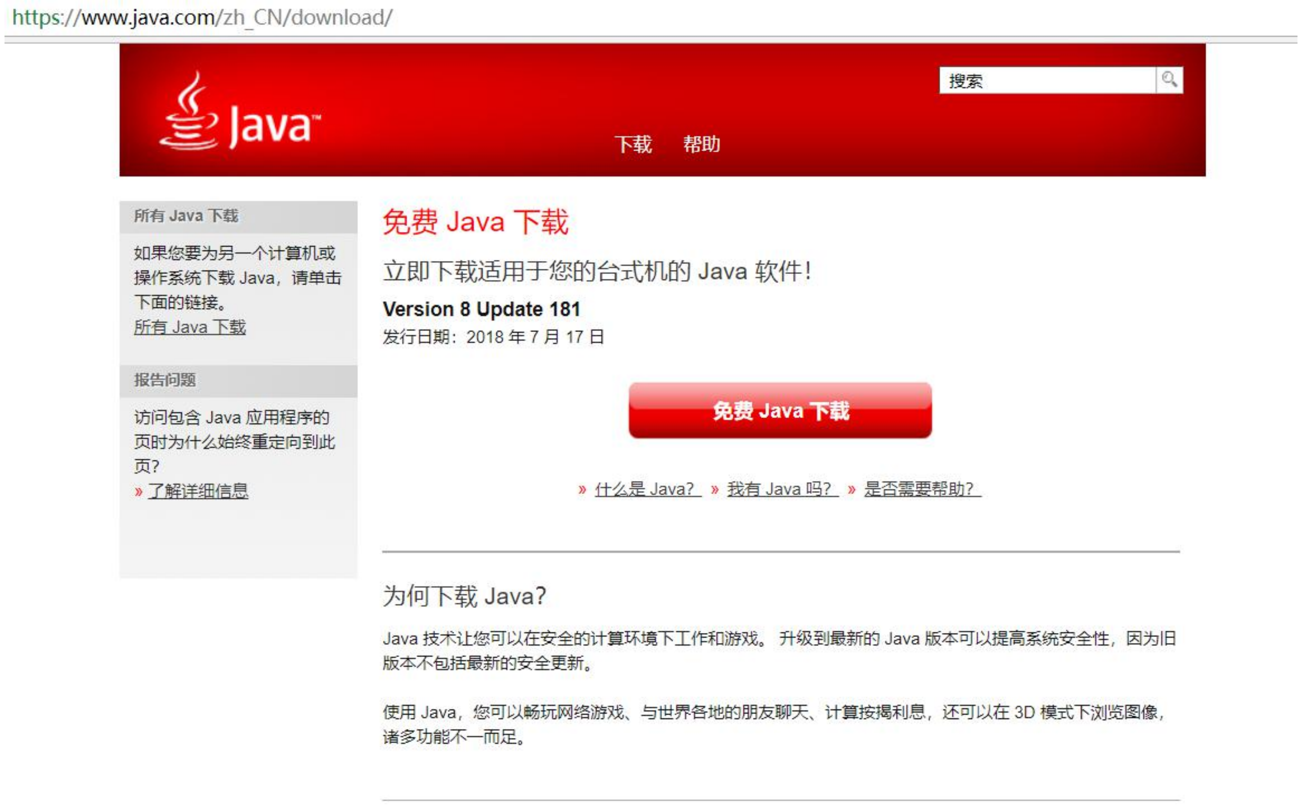
.java => .class => JVM execution



Cited from "junji zhi university of toronto"

Installing Java

- Downloading Java Development Kit (JDK) from Oracle


https://www.java.com/zh_CN/download/

所有 Java 下载

如果您要为另一个计算机或操作系统下载 Java，请单击下面的链接。
所有 Java 下载

报告问题

访问包含 Java 应用程序的页时为什么始终重定向到此页?
» 了解详细信息

免费 Java 下载

立即下载适用于您的台式机的 Java 软件!

Version 8 Update 181

发行日期：2018 年 7 月 17 日

» 什么是 Java? » 我有 Java 吗? » 是否需要帮助?

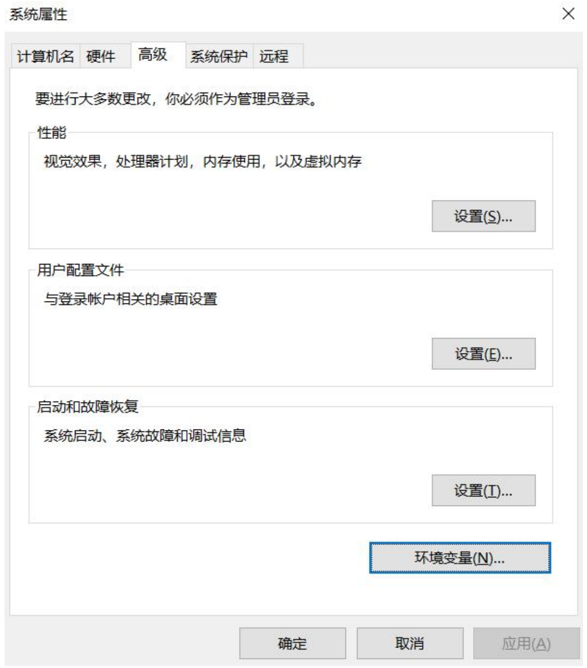
为何下载 Java?

Java 技术让您可以在安全的计算环境下工作和游戏。 升级到最新的 Java 版本可以提高系统安全性，因为旧版本不包括最新的安全更新。

使用 Java，您可以畅玩网络游戏、与世界各地的朋友聊天、计算按揭利息，还可以在 3D 模式下浏览图像，诸多功能不一而足。

Installing Java

- Setting JAVA_HOME (Windows):
  - E.g., *C:\Program Files\Java\jdk1.7.0_45*
- Setting **path** and **classpath**


系统属性

要进行大多数更改，你必须作为管理员登录。

性能

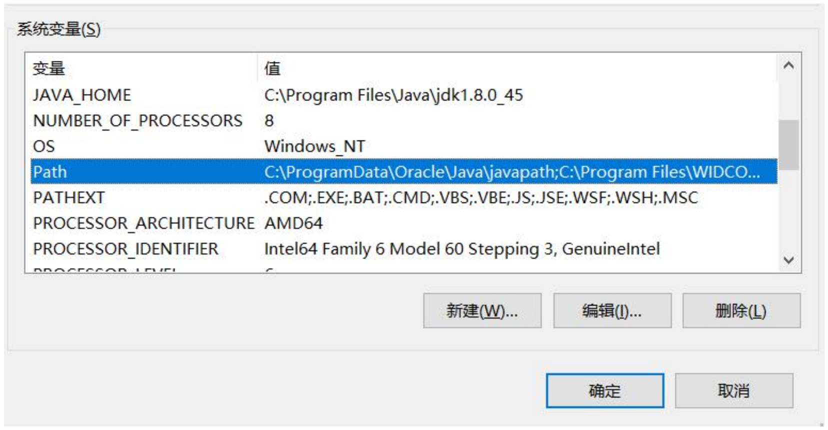
视觉效果，处理器计划，内存使用，以及虚拟内存

用户配置文件

与登录帐户相关的桌面设置

启动和故障恢复

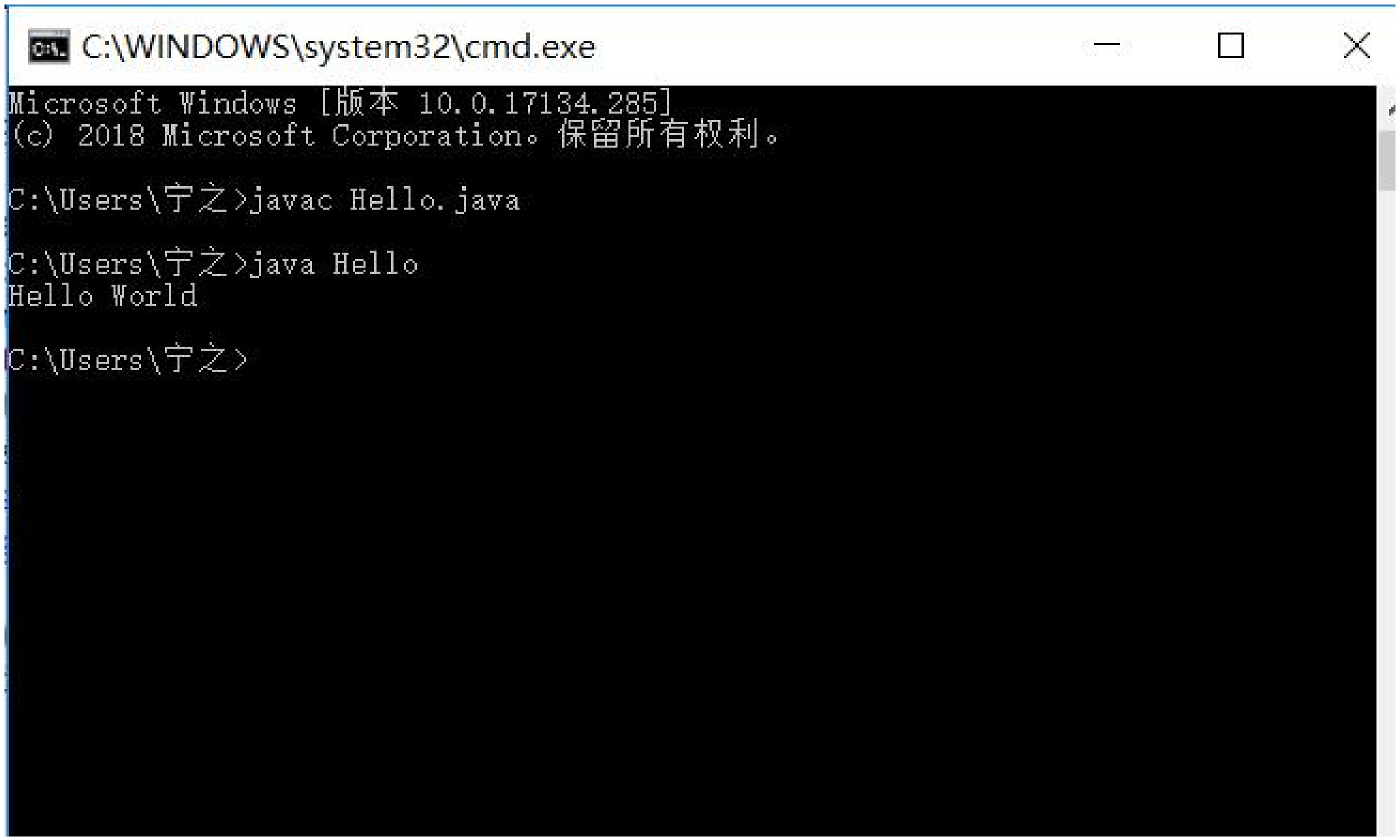
系统启动、系统故障和调试信息


系统变量(S)

变量	值
JAVA_HOME	C:\Program Files\Java\jdk1.8.0_45
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\ProgramData\Oracle\Java\javapath;C:\Program Files\WIDCO...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 60 Stepping 3, GenuineIntel

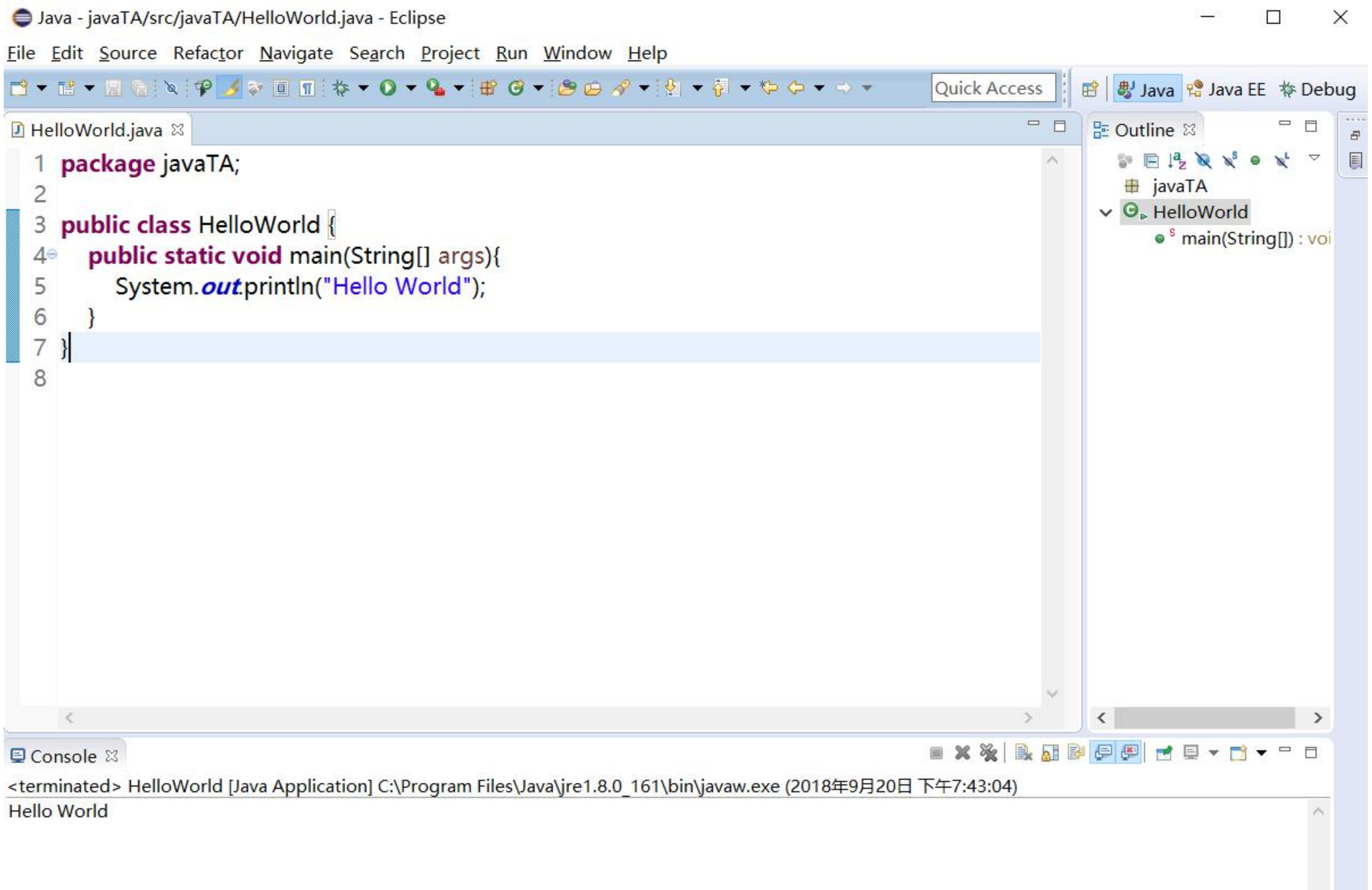
Compiling and Running in CMD

Compile .java File into a .class File (Command Line)



```
Microsoft Windows [版本 10.0.17134.285]
(c) 2018 Microsoft Corporation。保留所有权利。

C:\Users\宁之>javac Hello.java

C:\Users\宁之>java Hello
Hello World

C:\Users\宁之>
```

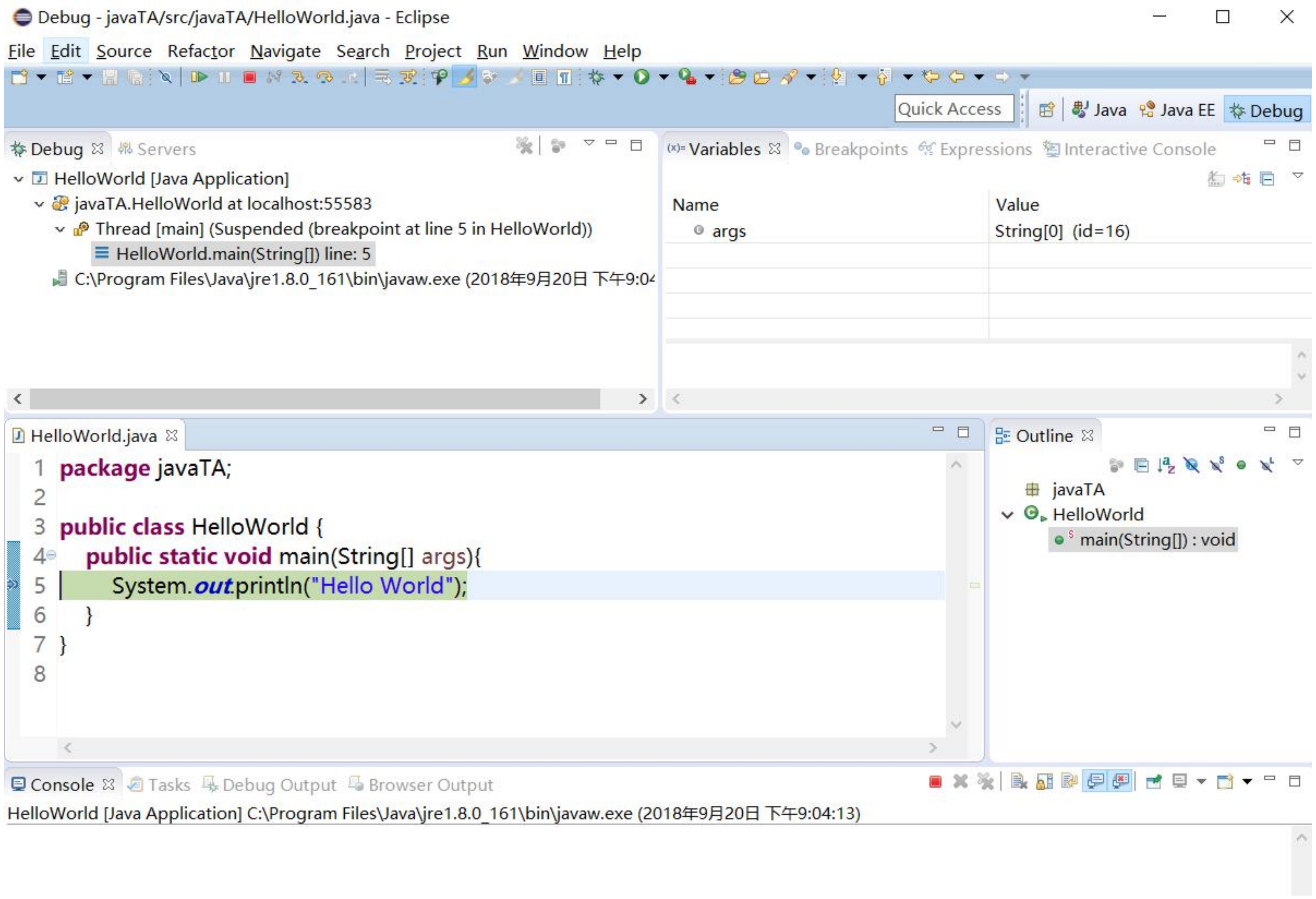
Running HelloWorld in Eclipse IDE


Java - javaTA/src/javaTA/HelloWorld.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Java EE
Debug
HelloWorld.java
Outline
javaTA
HelloWorld
main(String[]) : voi
Console

```java
package javaTA;

public class HelloWorld {
    public static void main(String[] args){
        System.out.println("Hello World");
    }
}
```

```
<terminated> HelloWorld [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (2018年9月20日 下午7:43:04)
Hello World
```

Debugging Java in Eclipse



HelloWorld [Java Application]
javaTA.HelloWorld at localhost:55583
Thread [main] (Suspended (breakpoint at line 5 in HelloWorld))
HelloWorld.main(String[]) line: 5
C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (2018年9月20日 下午9:0

Name	Value
args	String[0] (id=16)

```
package javaTA;

public class HelloWorld {
    public static void main(String[] args){
        System.out.println("Hello World");
    }
}

```

javaTA
HelloWorld
main(String[]) : void

HelloWorld [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (2018年9月20日 下午9:04:13)

Agenda

01 Introduction

02 Compiling, Running and Debugging

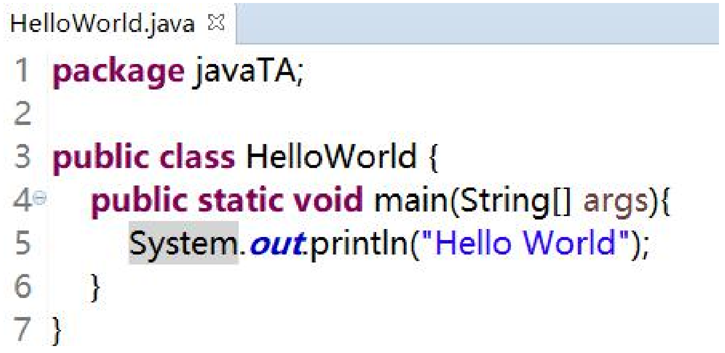
03 Java Language Syntax

04 Object-oriented

05 Threading

06 Program Example

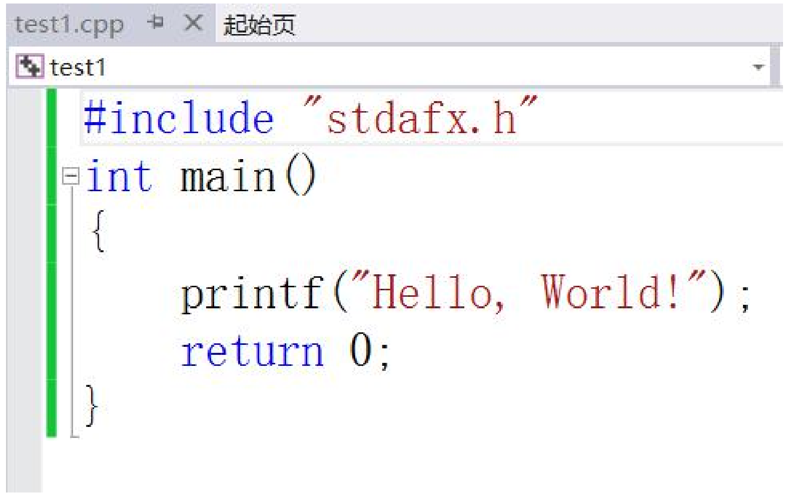
Example: Hello World Program



HelloWorld.java

```
package javaTA;

public class HelloWorld {
    public static void main(String[] args){
        System.out.println("Hello World");
    }
}
```

- Everything is in a class
- Compare with C



test1.cpp 起始页

test1

```
#include "stdafx.h"
int main()
{
    printf("Hello, World!");
    return 0;
}
```

Primitive Data Types

- **Primitive Data Types:** byte, short, int, long, float, double, boolean, char
- **Arrays** are also a class

```
long [] a = new  long[5];
```

  – You can get the length by visiting the length field of array object a, like this: **a.length**
- **String** class is very commonly used to represents character strings, for example

```
String s1 = "Hello ", s2 = "Wolrd!";
String s3 = s1 + s2;
```

Operators (same as C/C++)

- ++,-- Auto increment/decrement
- +,- Unary plus/minus
- *,/ Multiplication/division
- % Modulus
- +,- Addition/subtraction

Declaring Variables

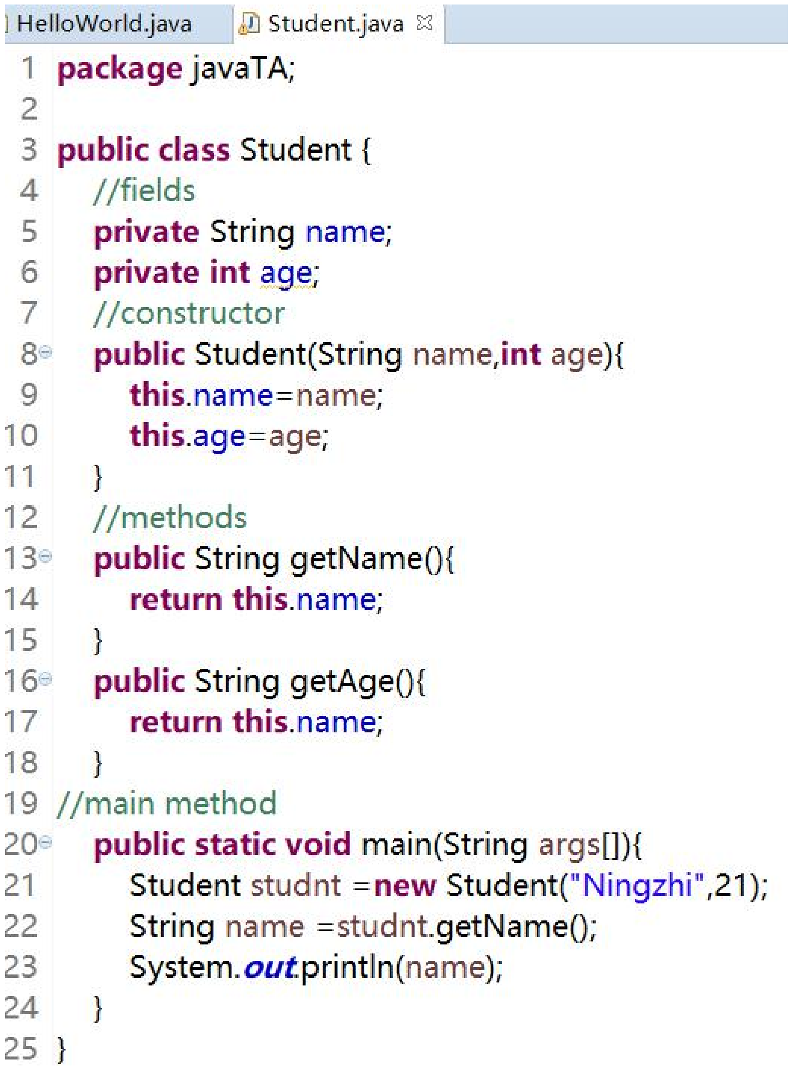
```
int n = 1;
char ch = ‘A’;
string s = “Hello”;
long L = new Long(100000);
boolean done = false;
final double pi =
3.14159265358979323846;
Employee joe = new Employee();
char [] a = new char[3];
Vector v = new Vector();
```

Compared with C/C++

- Java has no:
  - pointers
  - typedef
  - preprocessor
  - struct
  - unions
  - multiple inheritance
  - goto
  - operator overloading
  - malloc
  - …

Declaring a Class

- package
- class name
- constructor
- fields
- methods



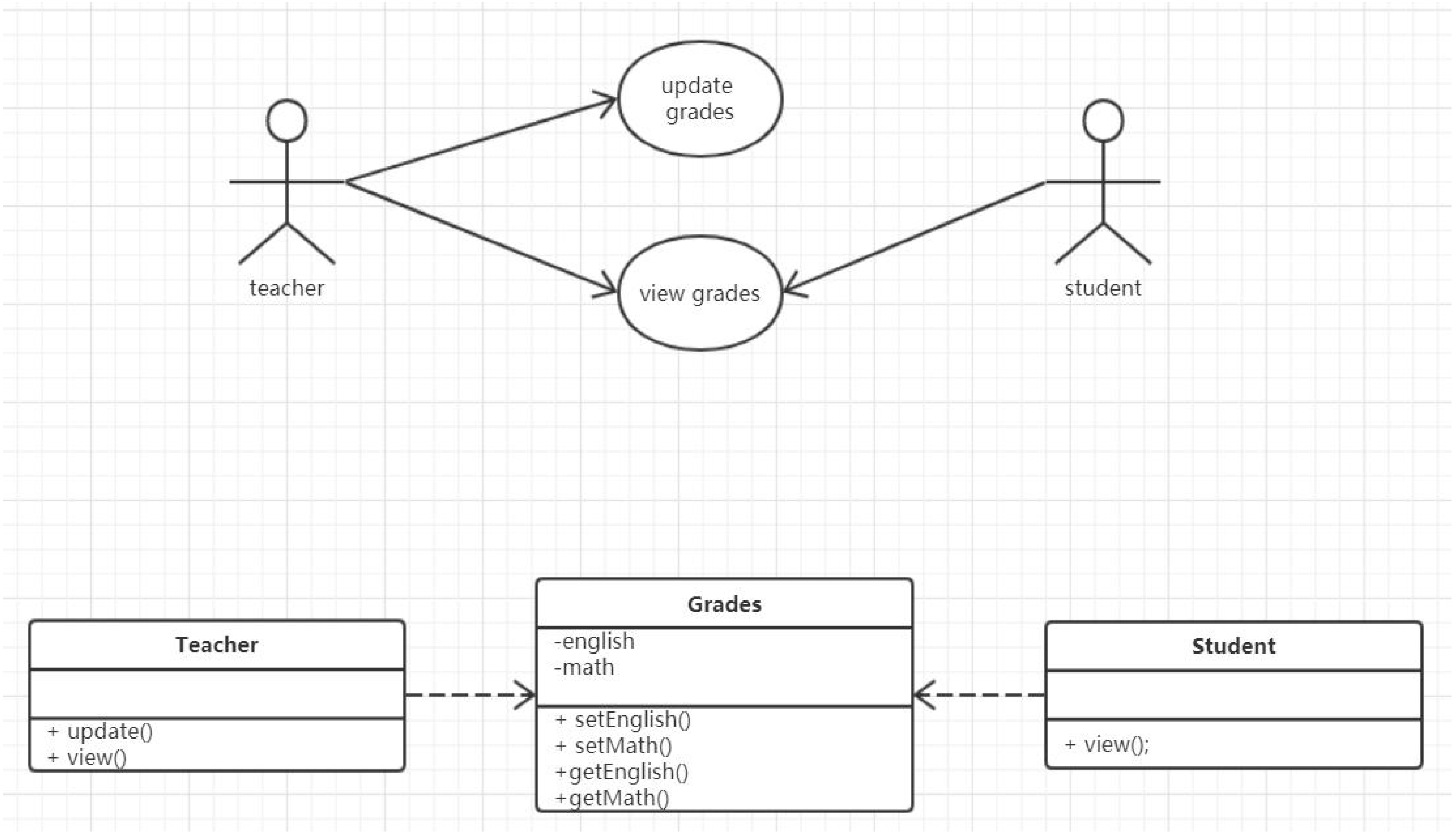
```java
package javaTA;

public class Student {
    //fields
    private String name;
    private int age;
    //constructor
    public Student(String name,int age){
        this.name=name;
        this.age=age;
    }
    //methods
    public String getName(){
        return this.name;
    }
    public String getAge(){
        return this.name;
    }
//main method
    public static void main(String args[]){
        Student studnt =new Student("Ningzhi",21);
        String name =studnt.getName();
        System.out.println(name);
    }
}
```

Agenda

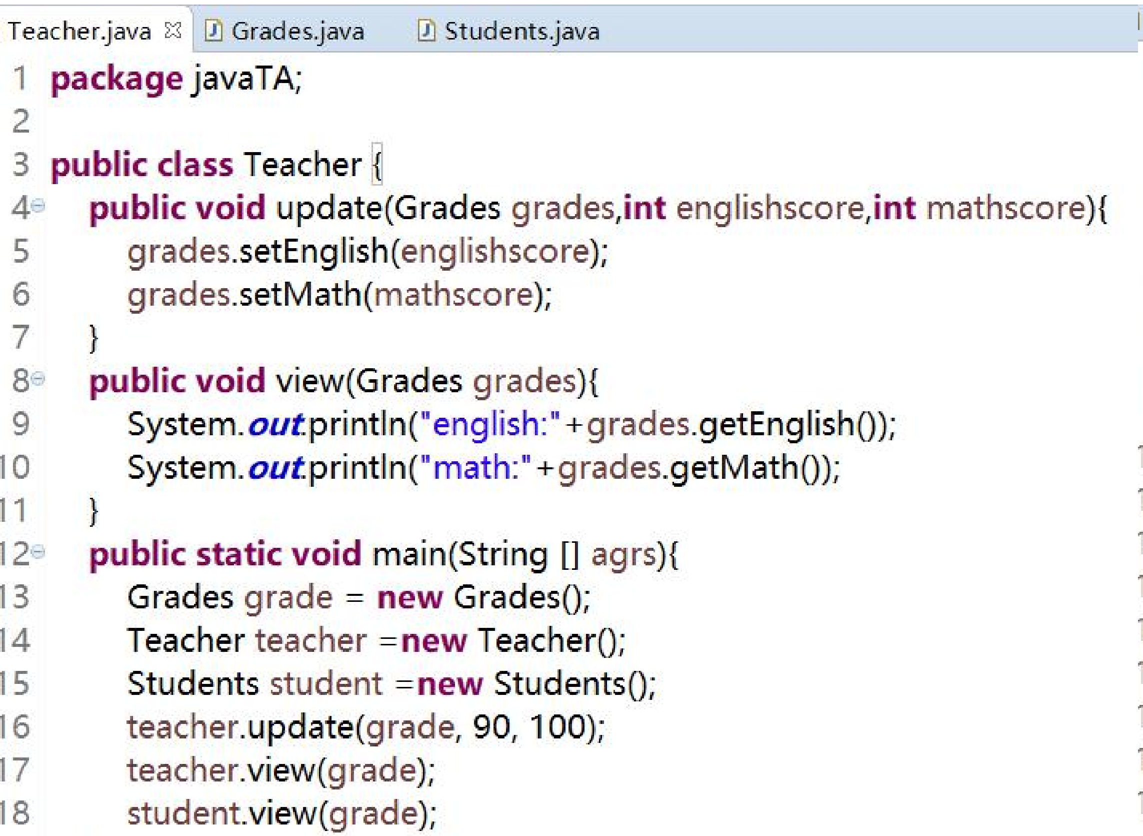
Object-oriented

- Advantages of object oriented programming
  - Modeling like human thinking mode
  - Easy to maintain
  - Easy to expand
  - Easy to reuse

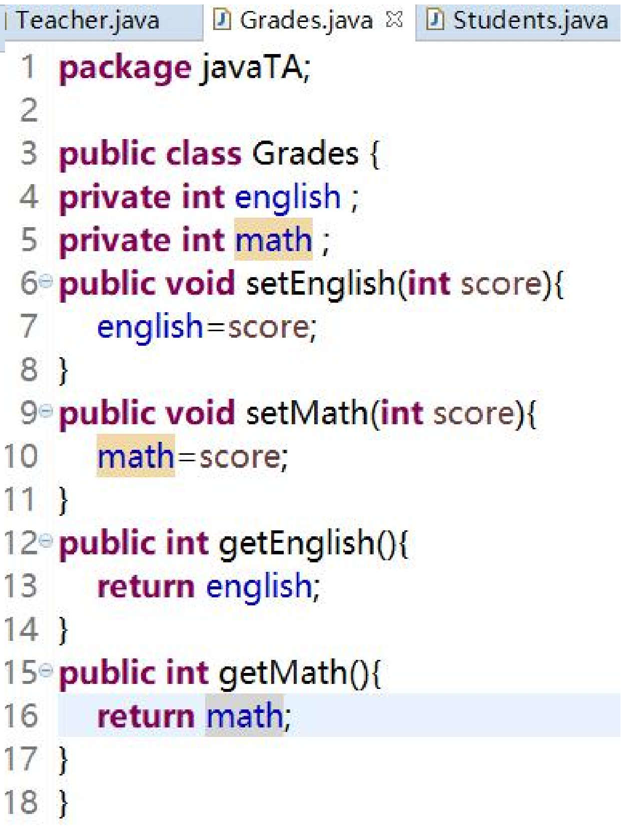
Object-oriented



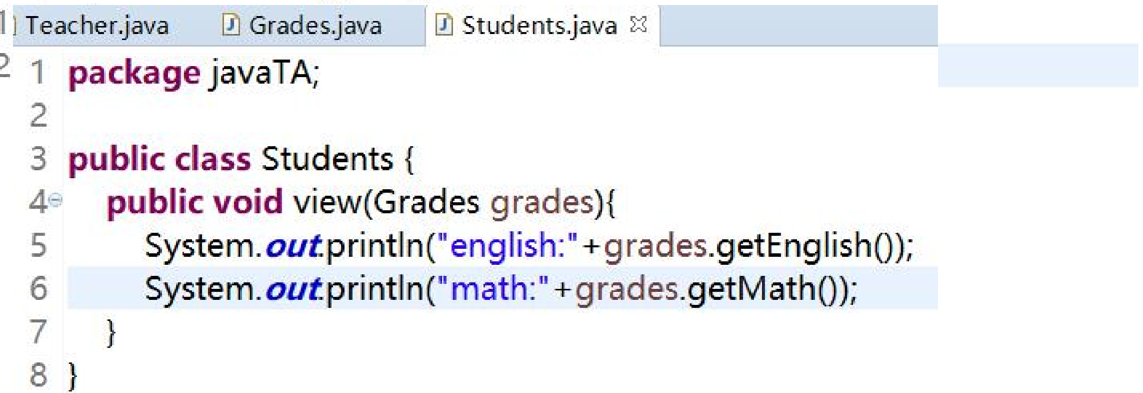
Object-oriented


Teacher.java | Grades.java | Students.java

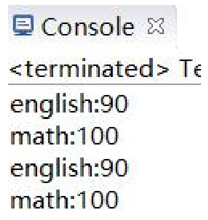
```java
package javaTA;

public class Teacher {
	public void update(Grades grades,int englishscore,int mathscore){
		grades.setEnglish(englishscore);
		grades.setMath(mathscore);
	}
	public void view(Grades grades){
		System.out.println("english:"+grades.getEnglish());
		System.out.println("math:"+grades.getMath());
	}
	public static void main(String [] agrs){
		Grades grade = new Grades();
		Teacher teacher =new Teacher();
		Students student =new Students();
		teacher.update(grade, 90, 100);
		teacher.view(grade);
		student.view(grade);
```


Teacher.java | Grades.java | Students.java

```java
package javaTA;

public class Grades {
private int english ;
private int math ;
public void setEnglish(int score){
	english=score;
}
public void setMath(int score){
	math=score;
}
public int getEnglish(){
	return english;
}
public int getMath(){
	return math;
}
}
```


Teacher.java | Grades.java | Students.java

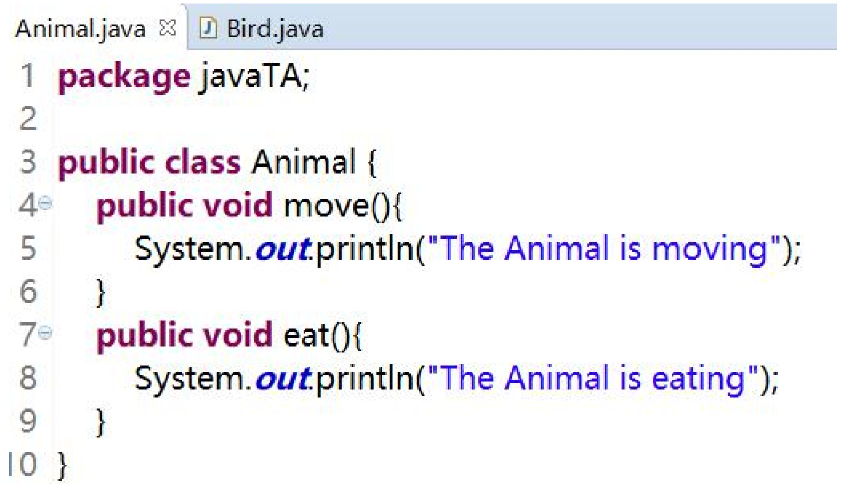
```java
package javaTA;

public class Students {
	public void view(Grades grades){
		System.out.println("english:"+grades.getEnglish());
		System.out.println("math:"+grades.getMath());
	}
}
```


Console

```
<terminated> Te
english:90
math:100
english:90
math:100
```

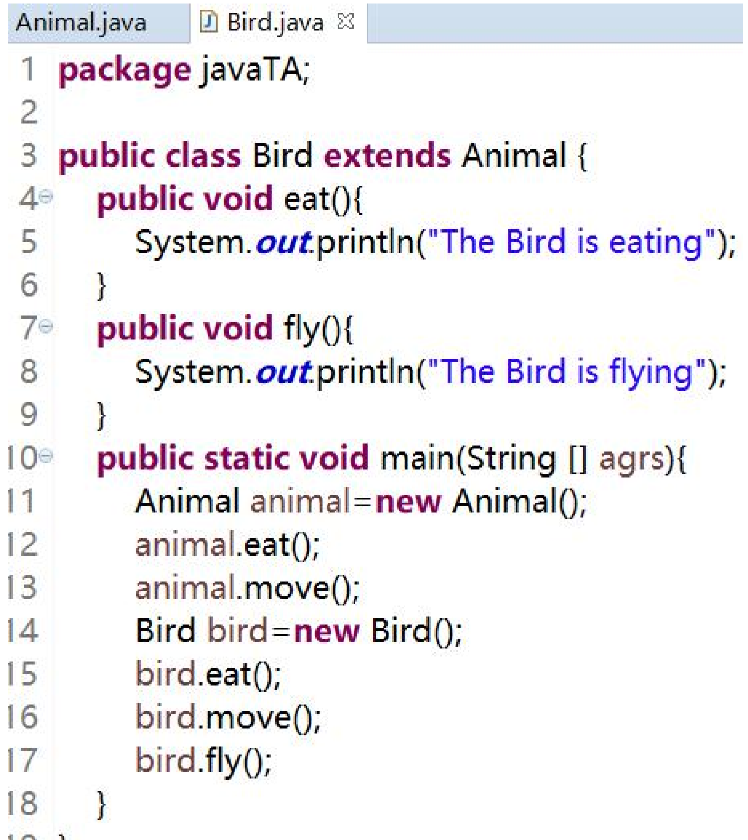
Inheritance in Java

- Java classes can be *derived* from other classes, thereby *inheriting* fields and methods from those classes.



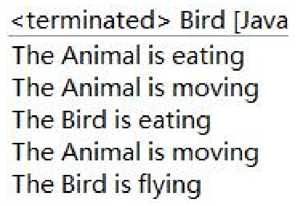
Animal.java | Bird.java

```java
package javaTA;

public class Animal {
    public void move(){
        System.out.println("The Animal is moving");
    }
    public void eat(){
        System.out.println("The Animal is eating");
    }
}
```



Animal.java | Bird.java

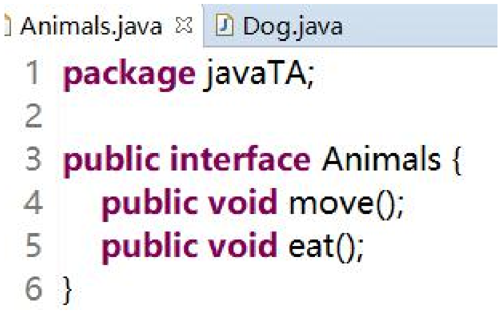
```java
package javaTA;

public class Bird extends Animal {
    public void eat(){
        System.out.println("The Bird is eating");
    }
    public void fly(){
        System.out.println("The Bird is flying");
    }
    public static void main(String [] agrs){
        Animal animal=new Animal();
        animal.eat();
        animal.move();
        Bird bird=new Bird();
        bird.eat();
        bird.move();
        bird.fly();
    }
```



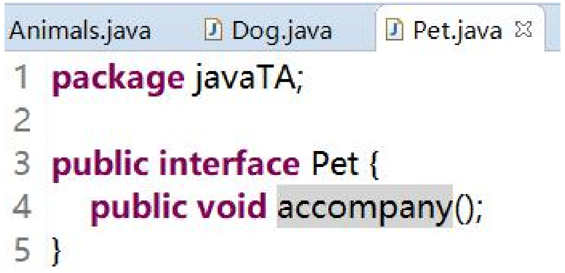
```
<terminated> Bird [Java
The Animal is eating
The Animal is moving
The Bird is eating
The Animal is moving
The Bird is flying
```

Interface

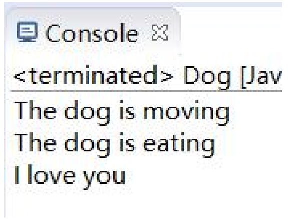
- all methods are abstract methods.
- one class can implement multiple interfaces.



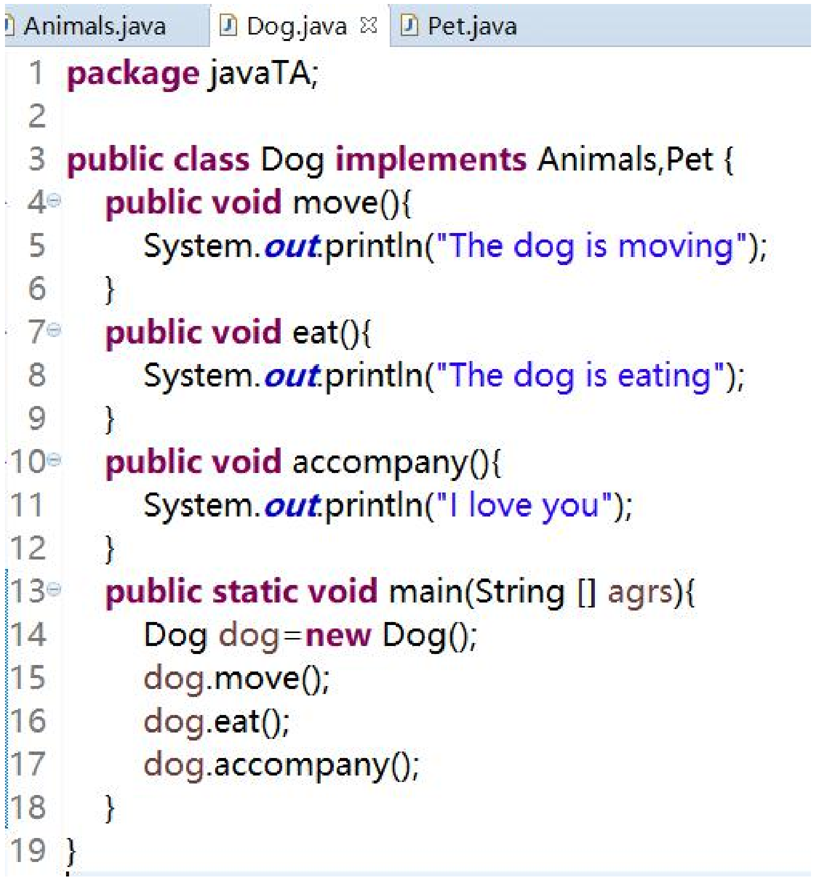
```java
package javaTA;

public interface Animals {
    public void move();
    public void eat();
}
```



```java
package javaTA;

public interface Pet {
    public void accompany();
}
```

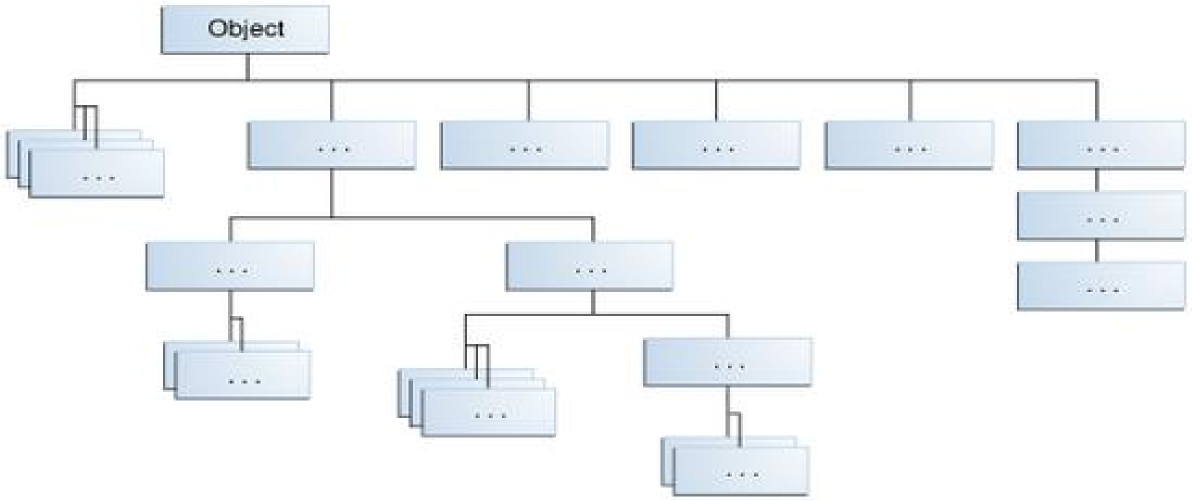


```
<terminated> Dog [Jav
The dog is moving
The dog is eating
I love you
```



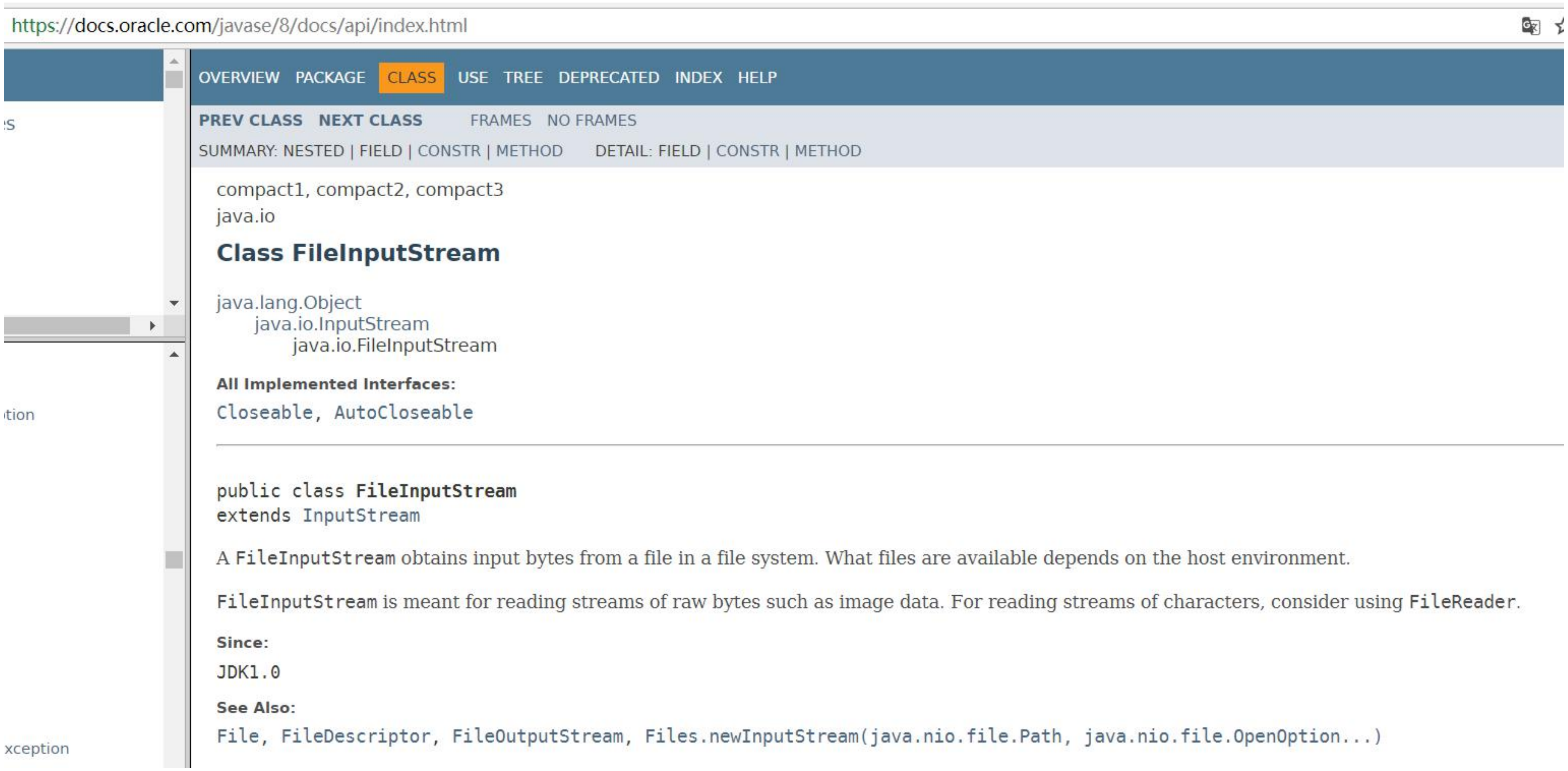
```java
package javaTA;

public class Dog implements Animals,Pet {
    public void move(){
        System.out.println("The dog is moving");
    }
    public void eat(){
        System.out.println("The dog is eating");
    }
    public void accompany(){
        System.out.println("I love you");
    }
    public static void main(String [] agrs){
        Dog dog=new Dog();
        dog.move();
        dog.eat();
        dog.accompany();
    }
}
```

Common Root: Object



Cited from "junji zhi university of toronto"

A Example: FileInputStream


https://docs.oracle.com/javase/8/docs/api/index.html

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP
PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD
:s
tion
xception

compact1, compact2, compact3
java.io

Class FileInputStream

java.lang.Object
    java.io.InputStream
        java.io.FileInputStream

All Implemented Interfaces:
`Closeable, AutoCloseable`

```
public class FileInputStream
extends InputStream
```

A `FileInputStream` obtains input bytes from a file in a file system. What files are available depends on the host environment.

`FileInputStream` is meant for reading streams of raw bytes such as image data. For reading streams of characters, consider using `FileReader`.

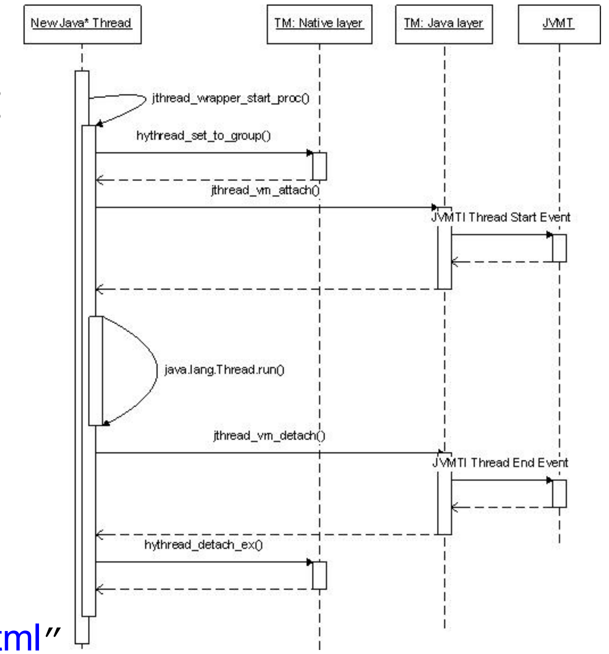
Since:
`JDK1.0`

See Also:
`File, FileDescriptor, FileOutputStream, Files.newInputStream(java.nio.file.Path, java.nio.file.OpenOption...)`

Agenda

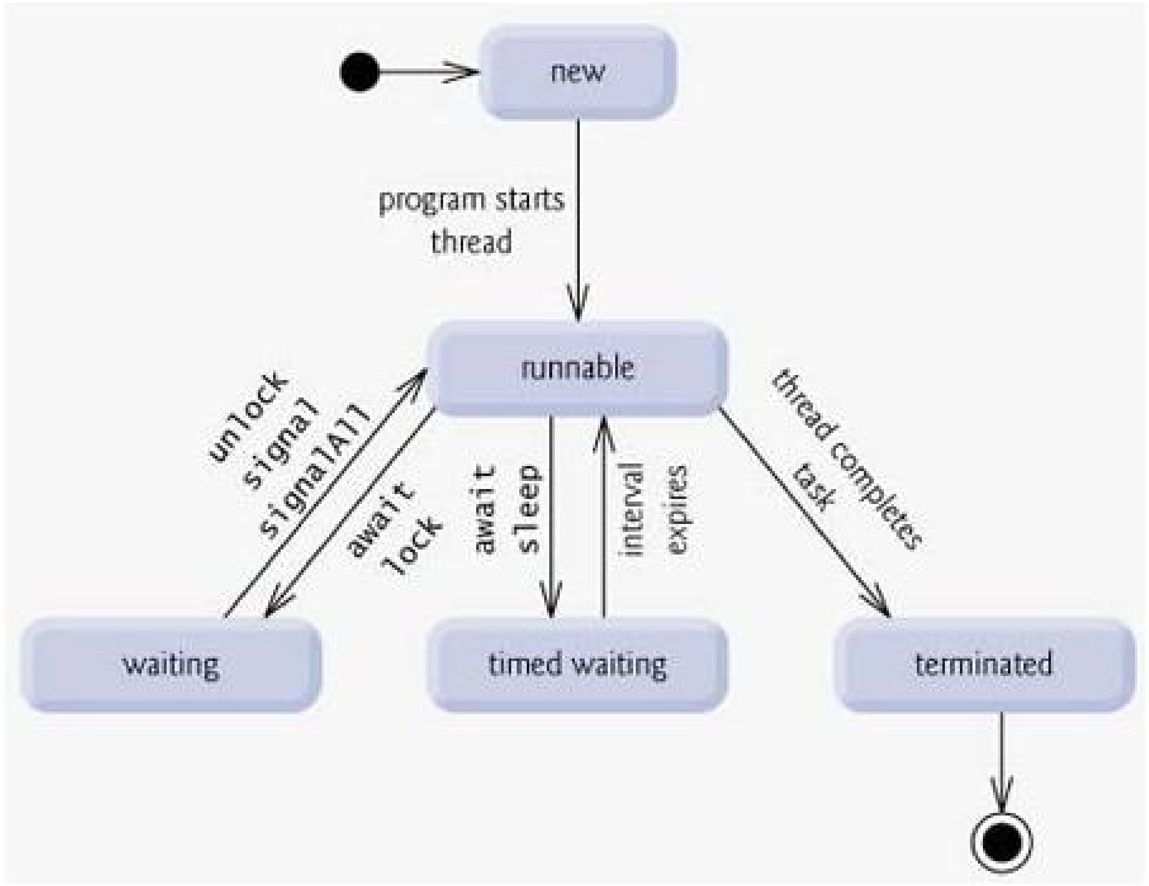
Java Threading

- A *thread* is a thread of execution in a program
- JVM allows an application to have multiple threads running concurrently.
- Apache Harmony example:



Cited from "http://harmony.apache.org/subcomponents/drlvm/TM.html"

Thread Lifecycle



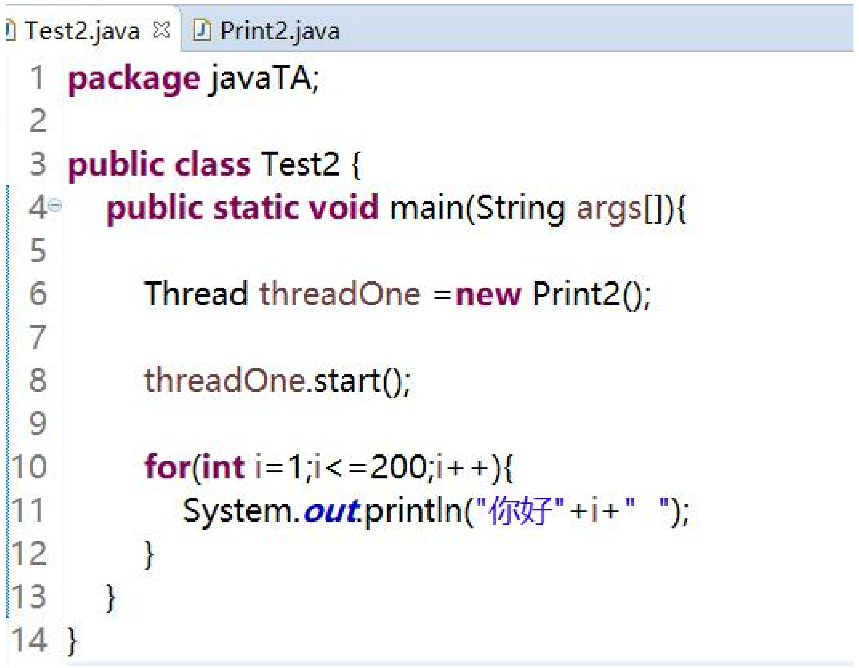
Cited from “junji zhi university of toronto”

Two Ways to do Threading

1. Extends Thread Class

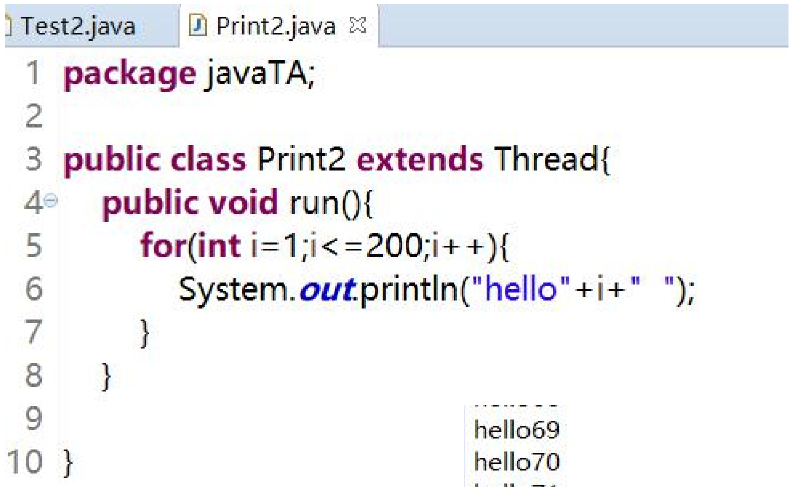
2. Implements Runnable Interface

Extends Thread class



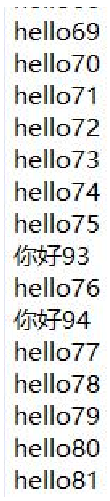
Test2.java

```java
package javaTA;

public class Test2 {
    public static void main(String args[]){

        Thread threadOne =new Print2();

        threadOne.start();

        for(int i=1;i<=200;i++){
            System.out.println("你好"+i+"  ");
        }
    }
}
```



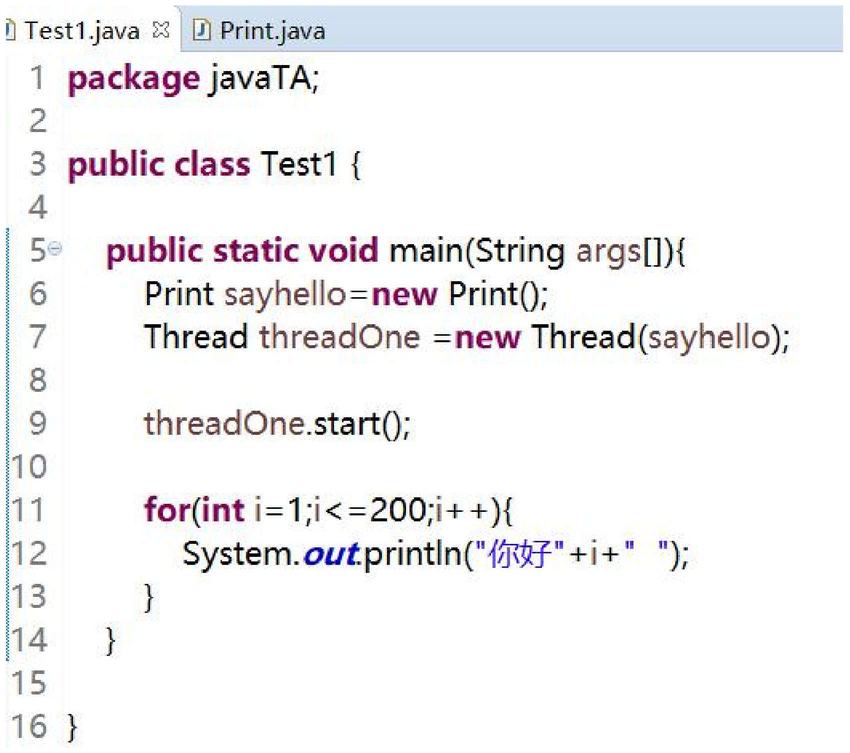
Print2.java

```java
package javaTA;

public class Print2 extends Thread{
    public void run(){
        for(int i=1;i<=200;i++){
            System.out.println("hello"+i+"  ");
        }
    }

}
```



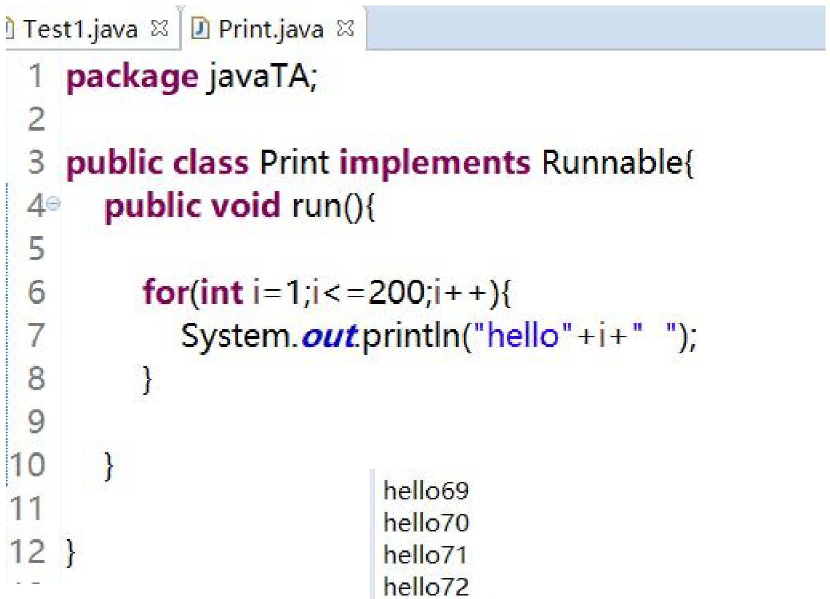
```
hello69
hello70
hello71
hello72
hello73
hello74
hello75
你好93
hello76
你好94
hello77
hello78
hello79
hello80
hello81
```

Implements Runnable interface



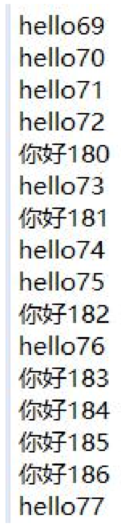
Test1.java ☒ Print.java

```java
package javaTA;

public class Test1 {

    public static void main(String args[]){
        Print sayhello=new Print();
        Thread threadOne =new Thread(sayhello);

        threadOne.start();

        for(int i=1;i<=200;i++){
            System.out.println("你好"+i+"  ");
        }
    }

}
```



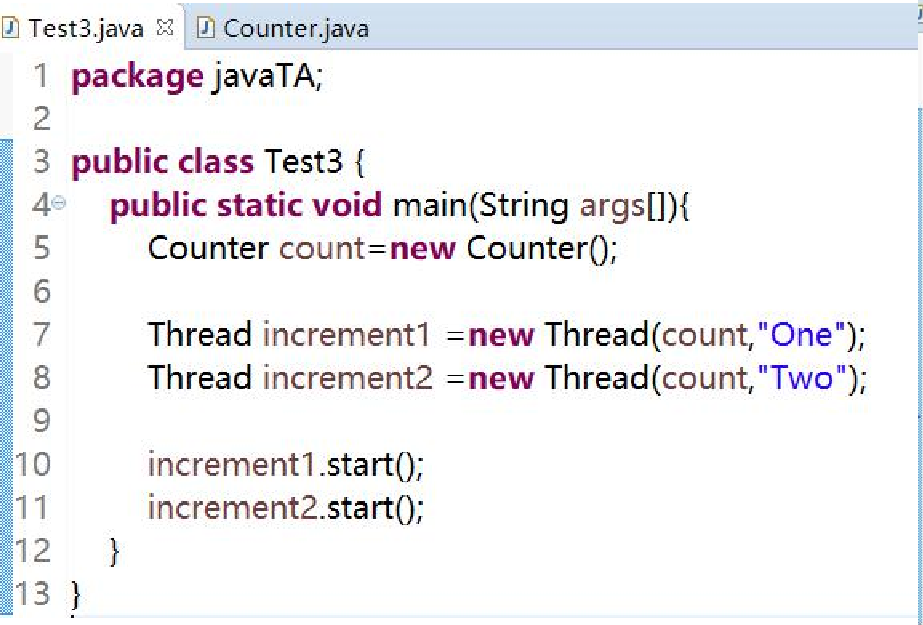
Test1.java ☒ Print.java ☒

```java
package javaTA;

public class Print implements Runnable{
    public void run(){

        for(int i=1;i<=200;i++){
            System.out.println("hello"+i+"  ");
        }

    }

}
```



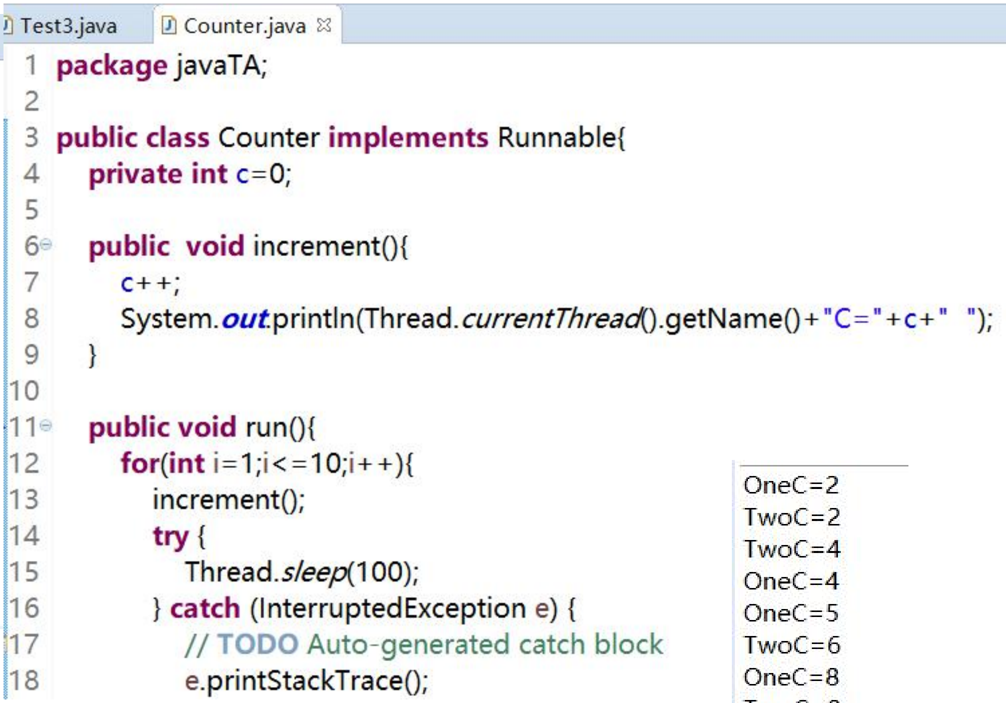
```
hello69
hello70
hello71
hello72
你好180
hello73
你好181
hello74
hello75
你好182
hello76
你好183
你好184
你好185
你好186
hello77
```

Thread Interference



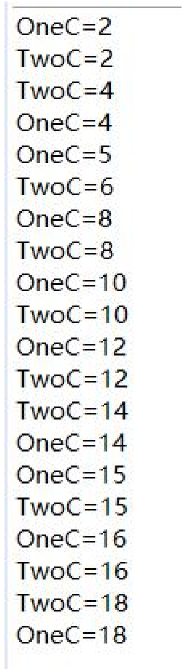
Test3.java Counter.java

```java
package javaTA;

public class Test3 {
    public static void main(String args[]){
        Counter count=new Counter();

        Thread increment1 =new Thread(count,"One");
        Thread increment2 =new Thread(count,"Two");

        increment1.start();
        increment2.start();
    }
}
```



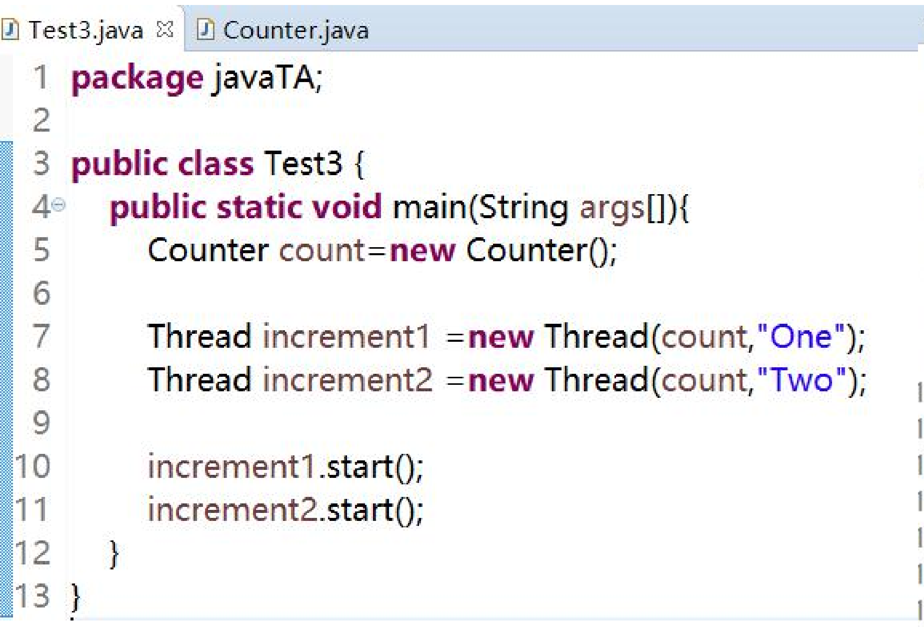
Test3.java Counter.java

```java
package javaTA;

public class Counter implements Runnable{
   private int c=0;

   public  void increment(){
      c++;
      System.out.println(Thread.currentThread().getName()+"C="+c+"  ");
   }

   public void run(){
      for(int i=1;i<=10;i++){
         increment();
         try {
            Thread.sleep(100);
         } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
```



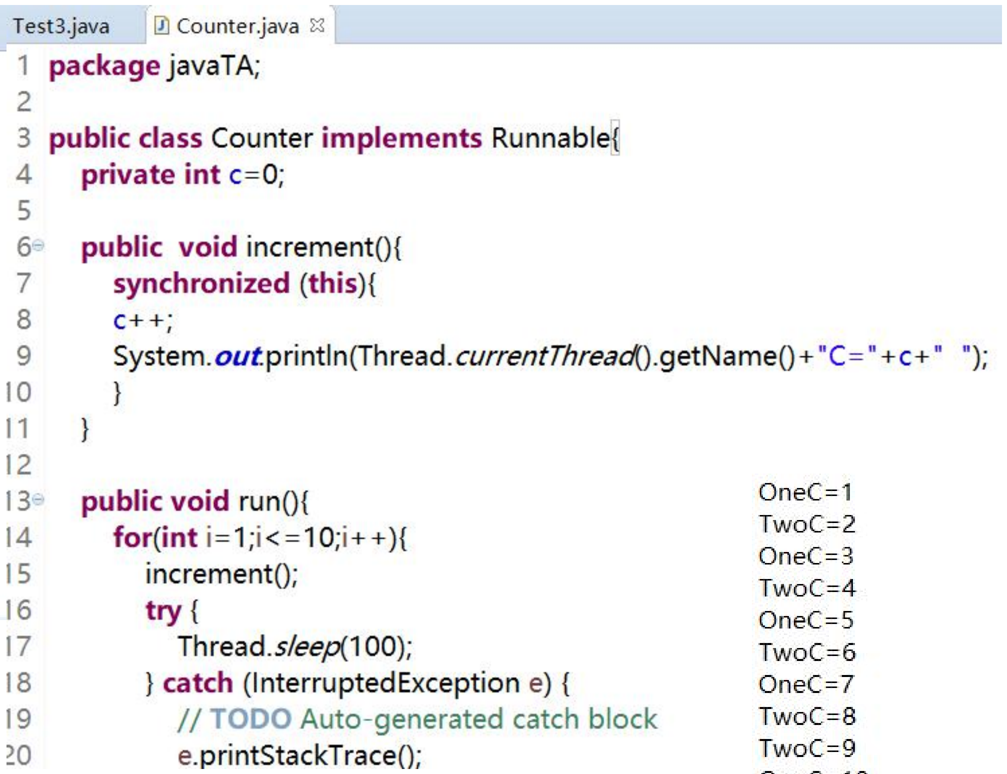
```
OneC=2
TwoC=2
TwoC=4
OneC=4
OneC=5
TwoC=6
OneC=8
TwoC=8
OneC=10
TwoC=10
OneC=12
TwoC=12
TwoC=14
OneC=14
OneC=15
TwoC=15
OneC=16
TwoC=16
TwoC=18
OneC=18
```

Synchronization



Test3.java Counter.java

```java
package javaTA;

public class Test3 {
    public static void main(String args[]){
        Counter count=new Counter();

        Thread increment1 =new Thread(count,"One");
        Thread increment2 =new Thread(count,"Two");

        increment1.start();
        increment2.start();
    }
}
```



Test3.java Counter.java

```java
package javaTA;

public class Counter implements Runnable{
    private int c=0;

    public  void increment(){
        synchronized (this){
        c++;
        System.out.println(Thread.currentThread().getName()+"C="+c+"  ");
        }
    }

    public void run(){
        for(int i=1;i<=10;i++){
            increment();
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
```

```
OneC=1
TwoC=2
OneC=3
TwoC=4
OneC=5
TwoC=6
OneC=7
TwoC=8
TwoC=9
OneC=10
OneC=11
TwoC=12
OneC=13
TwoC=14
TwoC=15
OneC=16
TwoC=17
OneC=18
OneC=19
TwoC=20
```

Agenda

Thanks!

Reference junji zhi.university of toronto.
Introduction to Java Programming Language