CSC3100 Data Structures
Lecture 2: A brief introduction to Java

Yixiang Fang
School of Data Science (SDS)
The Chinese University of Hong Kong, Shenzhen

Outline

- Why do we use Java to learn data structures?
- What we will learn and NOT learn about Java?
- Basic knowledge of Java
  - JDK/JVM/JRE
  - Keywords, declaration, expressions, class/object, method, others

Why do we choose Java?

- This course is not just about reading and writing — we need to do some practice by implementing data structures and algorithms

- Java is a multi-platform, object-oriented, and network-centric language
  - It is one of the most frequently used programming languages

- Used for
  - Developing Android Apps
  - Helps you to create Enterprise Software
  - Wide range of Mobile java Applications
  - Scientific Computing Applications
  - Use for Big Data Analytics
  - Java Programming of Hardware devices
  - Used for Server-Side Technologies like Apache, etc.
  - …
  - much more!

Why do we choose Java?

- Java features
  - It is one of the easy-to-use programming languages to learn
  - Java is platform-independent. Write code once and run it on almost any computing platform.
  - It is designed for building object-oriented applications
  - It is a multithreaded language with automatic memory management
  - It is created for the distributed environment of the Internet
  - …

Java vs C/C++

- Advantages
  - Easier to learn
  - No pointer, safer
  - Automatic memory management, including garbage collection
  - Cross platforms
  - More powerful standard libraries
  - …

- Here are the reasons why you should learn Java:
  - Java is very easy to learn
  - Java developers are in demand, and it easy to get a job as a Java programmer
  - Java is often used for research (e.g., big data processing platforms like Hadoop and Spark)
  - It has a good collection of open-source libraries
  - Java and Java-based IDEs are often provided free of charge

What we will learn about Java?

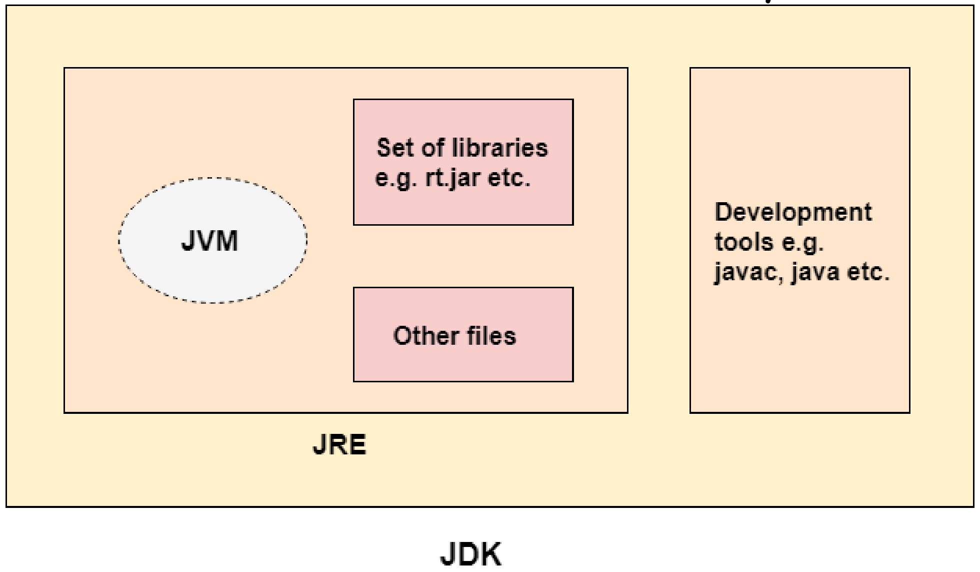
- What we will learn?
  - JDK, JVM, JRE
  - Keywords
  - Simple declarations
  - Statements
  - Classes/objects
  - Methods
  - Exceptions

- What we will NOT learn?
  - Interface
  - Abstract class
  - Inheritance
  - GUI
  - Multi-thread
  - Garbage collection
  - Network communication
  - Web design
  - Android Apps development
  - …
  - Many others

Only focus on the basic knowledge of Java that is used for implementing the data structures and algorithms in this course!

JDK, JVM, JRE

- Java Development Kit (JDK)
  - a software development environment which is used to develop Java applications and applets
- Java Runtime Environment (JRE)
  - provides the minimum requirements for executing a Java application; it consists of JVM, core classes, etc.
- JVM (Java Virtual Machine)
  - an abstract machine that doesn't physically exist, a specification that provides a runtime environment in which Java bytecode can be executed.


Set of libraries e.g. rt.jar etc.
JVM
Other files
Development tools e.g. javac, java etc.
JRE
JDK

Java Development kit (JDK)

- JDK is a software development environment used for making applets and Java applications. Java developers can use it on Windows, macOS, Solaris, and Linux. JDK helps them to code and run Java programs.

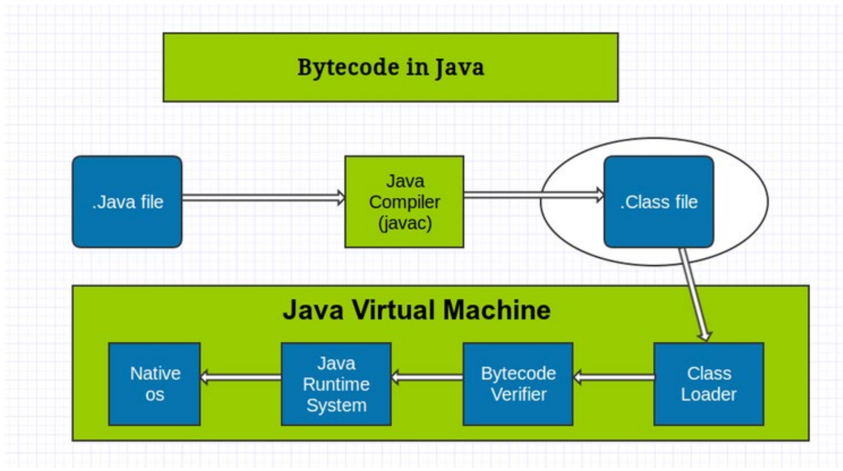
- Why use JDK?
  - JDK contains tools required to write Java programs and JRE to execute them.
  - It includes a compiler, Java application launcher, Appletviewer, etc.
  - Compiler converts code written in Java into byte code.
  - Java application launcher opens a JRE, loads the necessary class, and executes its main method.

Java Runtime Environment (JRE)

- JRE is a piece of software that is designed to run other software. It contains the class libraries, loader class, and JVM.
- In simple terms, if you want to run a Java program, you need JRE. If you are not a programmer, you don't need to install JDK, but just JRE to run Java programs.
- Why use JRE?
    - JRE contains class libraries, JVM, and other supporting files. It does not include any tool for Java development like a debugger, compiler, etc.
    - It uses important package classes like math, swing, util, lang, awt, and runtime libraries.
    - If you have to run Java applets, then JRE must be installed in your system.

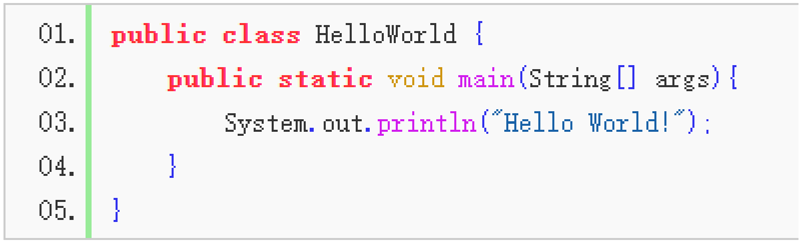
Java Virtual Machine (JVM)

- JVM is an engine that
  - provides a runtime environment to drive the Java Code or applications
  - converts Java bytecode into machine language. JVM is a part of the JRE
- Why use JVM?
  - JVM provides a platform-independent way of executing Java source code
  - It has numerous libraries, tools, and frameworks
  - Once you run a Java program, you can run on any platform and save lots of time
  - JVM comes with JIT (Just-in-Time) compiler that converts Java source code into low-level machine language. Hence, it runs faster than a regular application



Is JVM platform Independent or dependent?

The first Java program



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

- Source codes
  - Declare a class with name HelloWorld
  - Declare the main method
    - public static void main(String args[]){…}
  - Print "Hello World" to the console
    - System.out.println("Hello World")

(1) Java keywords

- Java keywords
  - Also known as reserved words
  - Keywords are particular words which acts as a key to a code
  - These are predefined words by Java so it cannot be used as a variable or object name

- Keywords of primitive types
  - int: used to declare a variable that can hold a 32-bit signed integer
  - boolean: used to declare a variable as a boolean type. It can hold True and False values only
  - double: is used to declare a variable that can hold a 64-bit floating-point numbers
  - char: used to declare a variable that can hold unsigned 16-bit Unicode characters
  - short: used to declare a variable that can hold a 16-bit integer
  - long: used to declare a variable that can hold a 64-bit integer
  - float: used to declare a variable that can hold a 32-bit floating-point number
  - byte: used to declare a variable that can hold an 8-bit data values

(1) Java keywords

- Keywords of loops
  - if: Java if keyword tests the condition. It executes the if block if condition is true.
  - else: used to indicate the alternative branches in an if statement.
  - continue: used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition.
  - break: used to break loop or switch statement. It breaks the current flow of the program at specified condition.

- Keywords of exceptions
  - try: used to start a block of code that will be tested for exceptions. The try block must be followed by either catch or finally block.
  - catch: used to catch the exceptions generated by try statements. It must be used after the try block only.
  - finally: used to create a block of code following a try block. The finally block always executes whether an exception occurs or not.

(1) Java keywords

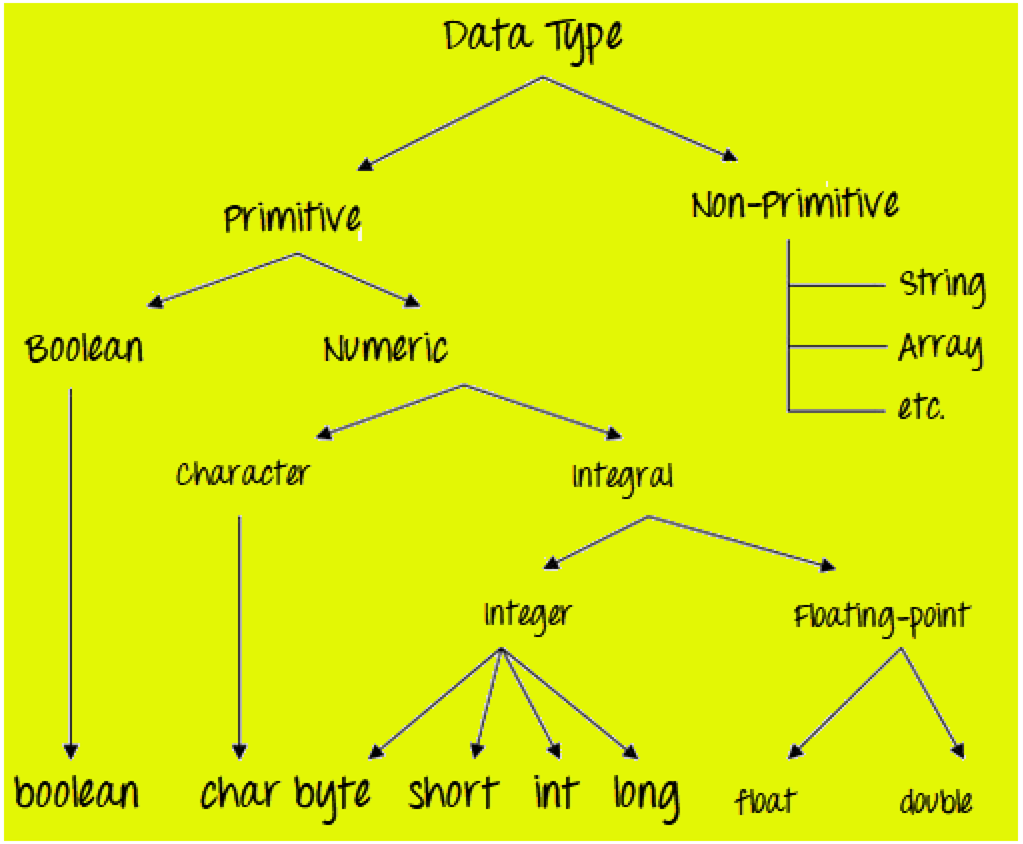
- Keywords of classes and objects
  - class: used to declare a class
  - new: used to create new objects

- Keywords of others
  - import: used to make classes and interfaces available and accessible to the current source code
  - package: used to declare a Java package that includes the classes
  - public: It is an access modifier. It is used to indicate that an item is accessible anywhere. It has the widest scope among all other modifiers
  - return: used to return from a method when its execution is complete
  - null: used to indicate that a reference does not refer to anything. It removes the garbage value

(2) Simple declaration

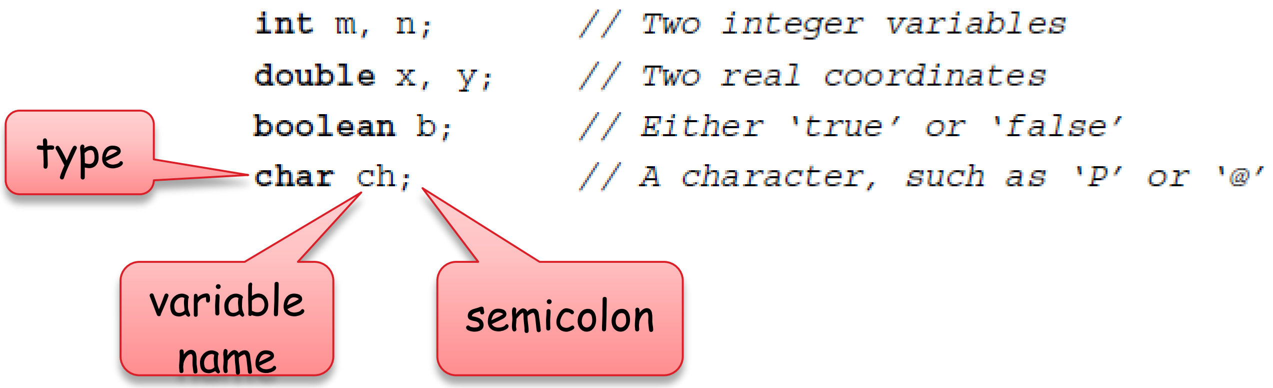
- Variable in Java is a data container that stores the data values during Java program execution
- Every variable is assigned data type which designates the type and quantity of value it can hold
- Variable is a memory location name of the data
- To use variables
  - Variable declaration
  - Variable initialization

(2) Simple declaration



(2) Simple declaration

- Variable declaration



```
int m, n;          // Two integer variables
double x, y;       // Two real coordinates
boolean b;         // Either 'true' or 'false'
char ch;           // A character, such as 'P' or '@'
```

(2) Simple declaration

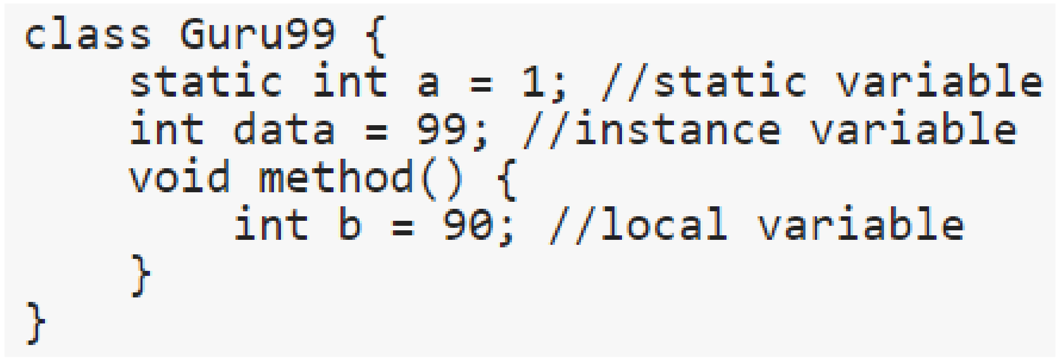
- Numeric expressions are written in much the same way as in other languages

```
n = 3 * (5 + 2);
x = y / 3.141592653;
n = m % 8;       // Modulo, i.e. n is now (m mod 8)
b = true;
ch = 'x';
```

- Division operator has two different things:

```
double f;
f = 1 / 3;             // f is now 0.0
f = 1.0 / 3.0;        // f is now 0.33333333...
```

(2) Simple declaration



```
class Guru99 {
    static int a = 1; //static variable
    int data = 99; //instance variable
    void method() {
        int b = 90; //local variable
    }
}
```

- Three types of variables
  - Local variables are declared inside the body of a method
  - Instance variables are defined without the STATIC keyword .They are defined Outside a method declaration. They are Object specific and are known as instance variables.
  - Static variables are initialized only once, at the start of the program execution. These variables should be initialized first, before the initialization of any instance variables.

(2) Simple declaration

- Type conversion (casting)
  - Assign a real value to an integer value: need a cast

```
double radians;
int degrees;
...
degrees = radians * 180 / 3.141592653;            // Error
degrees = (int) (radians * 180 / 3.141592653);    // OK
```

  - Assigning an integer value to a real variable without case is ok

(2) Simple declaration

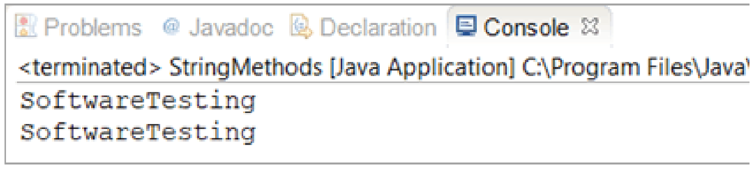
- A string is a series of characters, or an array of characters

```
//String is an array of characters
char[] arrSample = {'R', 'O', 'S', 'E'};
String strSample_1 = new String (arrSample);
```

- Use String class to handle strings

```
package codes;
import java.lang.String;

public class StringMethods {

    public static void main(String[] args) {

        String str1 = "Software";
        String str2 = "Testing";
        System.out.println(str1 + str2);
        System.out.println(str1.concat(str2));
    }
}
```

Output:



```
<terminated> StringMethods [Java Application] C:\Program Files\Java\
SoftwareTesting
SoftwareTesting
```

(3) Statements

- Statements can be grouped in blocks using "{ }"
- If and if-else statements

```
if (n == 3)
    x = 3.2;
```

Note:

- There is no `then` keyword
- The condition must be of boolean type and written within parentheses
- Comparison is made using '`==`'

  - Comparison operators: >, <, ==, >=, <=, !=

```
if (x != 0)
    y = 3.0 / x;        // Executed when x is non-zero
else
    y = 1;              // Executed when x is zero
```

(3) Statements

- More about Boolean expressions

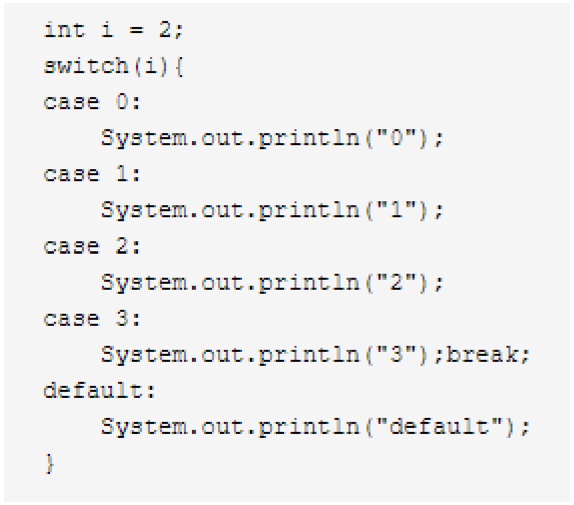
and	&&
or	||
not	!

- For example

```
int x, y;
boolean b;
...
if ((x <= 9 || y > 3) && !b) {
    b = true;
}
```

(3) Statements

- switch-case statement
  - The switch statement is a multi-way branch statement.
  - It provides an easy way to dispatch execution to different parts of code based on the value of the expression.
  - Basically, the expression can be byte, short, char, and int primitive data types.



```
int i = 2;
switch(i){
case 0:
    System.out.println("0");
case 1:
    System.out.println("1");
case 2:
    System.out.println("2");
case 3:
    System.out.println("3");break;
default:
    System.out.println("default");
}
```

(3) Statements

- while loop statement
  - Need to specify the stopping criterion

```
// Calculate exp(1). End when the term is less than 0.00001
double sum = 0.0;
double term = 1.0;
int k = 1;
while (term >= 0.00001) {
    sum = sum + term;
    term = term / k;
    k++;                    // Shortcut for 'k = k + 1'
}
```

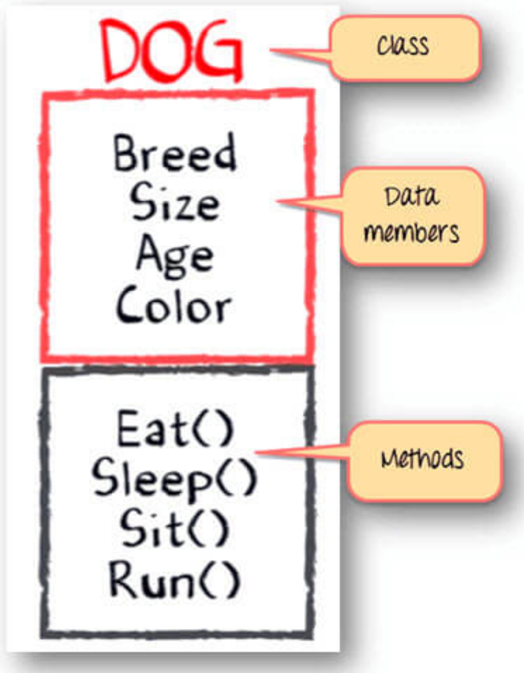
(3) Statements

- for loop statement
  - Initial value of value i
  - Stopping criterion of the loop
  - How to change the value of i in each iteration

```
// Calculate 1 + (1/2) + (1/3) + ... + (1/100)
int i;
double sum = 0.0;
for (i = 1; i <= 100; i++) {
    sum = sum + 1.0 / i;
}
```

(4) Class/object

- A class is a blueprint or prototype that defines the variables and the methods (functions) common to all Java objects of a certain kind
- An object is a specimen of a class
  - An object is an instance of a class
  - Software objects are often used to model real-world objects you find in everyday life



(4) Class

- A class declaration contains
  - A set of attributes (called instance variables)
  - A set of functions (called methods in Java)

```
class Turtle {
    private boolean penDown;
    protected int x, y;

    // Declare some more stuff
}
```

(4) Class

- Method
  - General methods:
    - public void jumpTo(int newX, int newY) {...}
  - Constructor methods
  - Main methods
    - public static void main(String [] args){…}

(4) Class

- Java constructor
  - It is a special method that is used to initialize a newly created object and is called just after the memory is allocated for the object
  - It can be used to initialize the objects to desired values or default values at the time of object creation
  - It is not mandatory for the coder to write a constructor for a class
- Rules for creating a java constructor
  - It has the same name as the class
  - It should not return a value not even void

(4) Class

- Methods

```
class Turtle {
    // Attribute declarations, as above

    public void jumpTo(int newX, int newY) {
        x = newX;
        y = newY;
    }

    public int getX() {
        return x;
    }
}
```

```
public Turtle(int initX, int initY) {
    x = initX;
    y = initY;
    penDown = false;
}
```

(4) Class

- Access modifiers
  - The public keyword is used to declare that something can be accessed from other classes
  - The protected keyword specifies that something can be accessed from within the class and all its subclasses, but not from the outside
  - When we do not mention any access modifier, it is called default access modifier. The scope of this modifier is limited to the package only.
  - The private declaration means that those attributes cannot be accessed outside of the class. In general, attributes should be kept private to prevent other classes from accessing them directly

Modifier	Class	Package	Subclass	Global
Public	Yes	Yes	Yes	Yes
Protected	Yes	Yes	Yes	No
Default	Yes	Yes	No	No
Private	Yes	No	No	No

(4) Class

- In Java, statements can only be written within methods in classes
  - This means that there must be some method which is called by the system when the program starts executing.
  - This method is called main and must be declared in the class which is started from the command line

(4) Class

- A main method usually creates a few objects and does some small work to get things going

```
public static void main(String[] args) {
    Turtle t = new Turtle(100, 200);
    t.right(90);
    while (t.getX() < 150) {
        t.forward(2);
    }
}
```

- Notes
  - static keyword: when the main method is called, it is not associated with an object, but with the class
  - The parameter args: if the Java interpreter is given any more information than the class name, this data is passed on to the main method in this parameter

(4) Class

- Examples

```java
public class ConfunDemo3 {
    public static void main(String[] args){
        Person z=new Person("zhangsan",3);
        z.show();
    }
}

class Person{
    private String name;
    private int age;
    public Person(String n,int m){
        name=n;
        age=m;
    }
    //getter
    public String getName(){
        return name;
    }
    public int getAget(){
        return age;
    }
    public void show(){
        System.out.println(name+"\n"+age);
    }
}
```

(5) Exception

- Many things can go wrong during the execution of a program
  - Faults introduced by the programmer, such as division by zero or calling a method with a null reference

  - Things out of the program's control, such as a user entering a garbage on the keyboard when the program expects a positive integer

(5) Exception

- Throw exceptions
  - Consider a method to read a positive integer from the keyboard could
  - What if the input character is not an integer?

```
public int getNatural() throws IOException {
    char ch;
    while (more input) {
        ch = (read character);
        if (ch < '0' || ch > '9') {
            throw new IOException("bad natural number");
        }
        ...
    }
    ...
}
```

(5) Exception

- Catch exceptions
  - The statement(s) within the try clause are executed as usual, but whenever an exception occurs, the try clause is interrupted and the statements within the corresponding catch clause are executed.

```
int m, n;
try {
    n = getNatural();
    m = n * 2; // If an exception is thrown, this is not executed
}
catch (IOException e) {
    // The user entered something wrong. Use 1 as default.
    n = 1;
    m = 2;
}
```

(6) Others

- Print something: writing to the console
  - System.out.print(xxx)
  - System.out.println(xxx)
  - Note: in eclipse, to write System.out.println(xxx), we can use shortcut keys: “syso”

```
System.out.print("Jag vill bo ");
System.out.println("i en svamp");
System.out.println("Annars får jag kramp");
```

The resulting output is:

```
Jag vill bo i en svamp
Annars får jag kramp
```

Variable values can be printed like this:

```
int a;
a = 6 * 7;
System.out.println("6 * 7 = " + a);
```

(6) Others

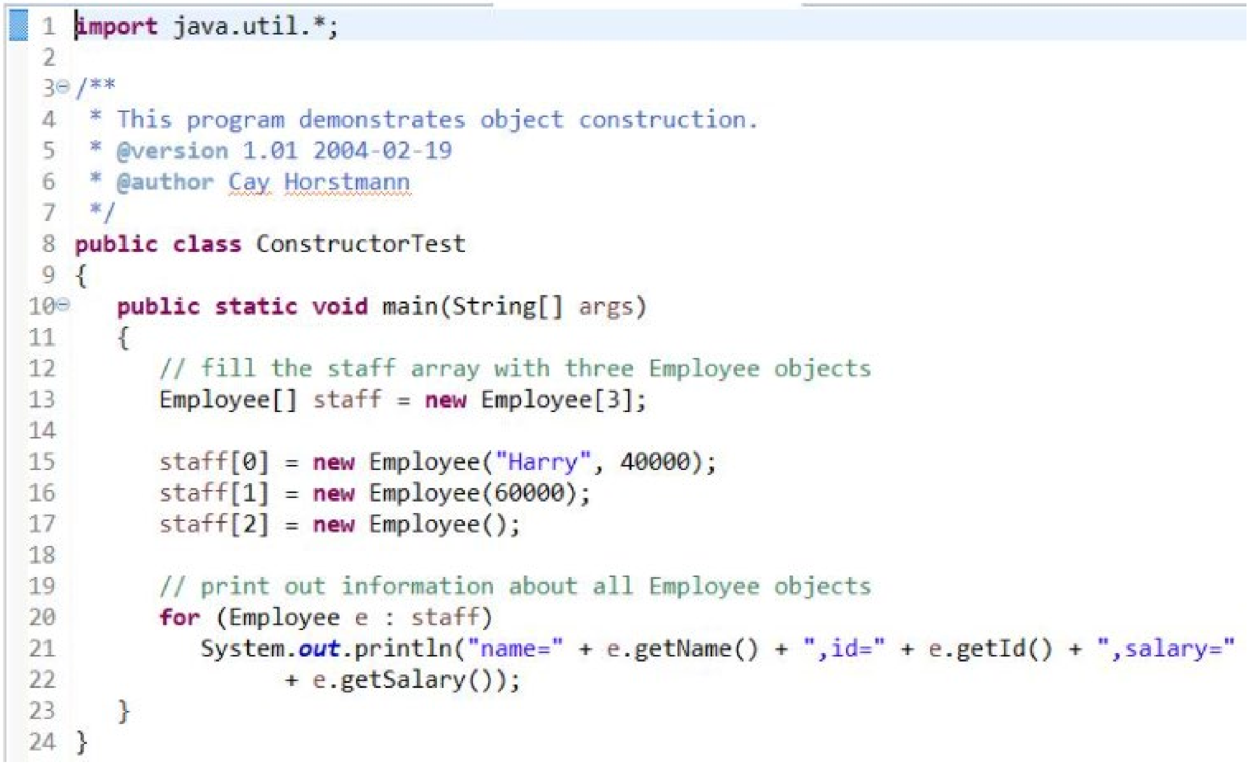
- Packages
  - **Package in Java** is a collection of classes, sub-packages, and interfaces. It helps organize your classes into a folder structure and make it easy to locate and use them.
  - More importantly, it helps improve code reusability.

```
import java.awt.*;
```

- Note: in eclipse, to import packages automatically, we can use shortcut keys: "Shit" + "Ctrl" + "o"

(6) Others

- Comments
  - Single sentence: two forward slashes //
  - A block of codes: /* xxx */



```java
import java.util.*;

/**
 * This program demonstrates object construction.
 * @version 1.01 2004-02-19
 * @author Cay Horstmann
 */
public class ConstructorTest
{
   public static void main(String[] args)
   {
      // fill the staff array with three Employee objects
      Employee[] staff = new Employee[3];

      staff[0] = new Employee("Harry", 40000);
      staff[1] = new Employee(60000);
      staff[2] = new Employee();

      // print out information about all Employee objects
      for (Employee e : staff)
         System.out.println("name=" + e.getName() + ",id=" + e.getId() + ",salary="
               + e.getSalary());
   }
}
```

(6) Others

- Array
  - Declaration

```
int[] someInts;          // An integer array
Turtle[] turtleFarm;     // An array of references to Turtles
```

  - Initialization

```
someInts = new int[30];
turtleFarm = new Turtle[100];
```

  - Use arrays: 0 ~ size-1

```
int i;
for (i = 0; i < someInts.length; i = i + 1) {
    someInts[i] = i * i;
}
```

What's the maximum number of elements in a Java array?

(5) Others

- Array of objects

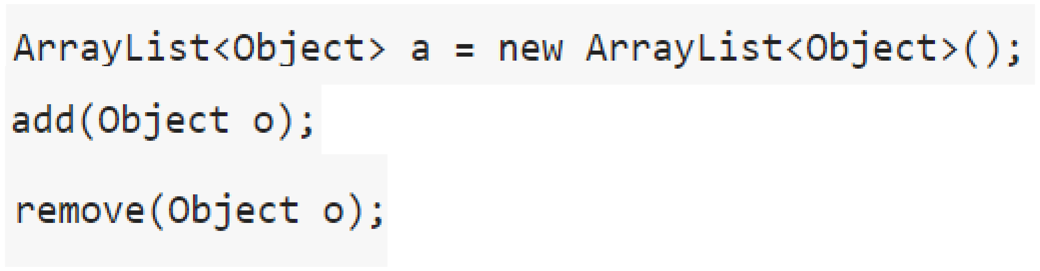
```
class ObjectArray{
   public static void main(String args[]){
     Account obj[] = new Account[2] ;
     //obj[0] = new Account();
     //obj[1] = new Account();
    obj[0].setData(1,2);
    obj[1].setData(3,4);
    System.out.println("For Array Element 0");
    obj[0].showData();
    System.out.println("For Array Element 1");
     obj[1].showData();
  }
}
class Account{
  int a;
  int b;
 public void setData(int c,int d){
   a=c;
   b=d;
 }
 public void showData(){
   System.out.println("Value of a ="+a);
   System.out.println("Value of b ="+b);
 }
}
```

This is a class

(6) Others

- ArrayList is a data structure that can
  - be stretched to accommodate additional elements within itself
  - shrink back to a smaller size when elements are removed.

- Notes
  - It is a very important data structure useful in handling the dynamic behavior of elements.
  - Although it provides more flexibility, it may take more space cost than an array, especially when the array is fully used

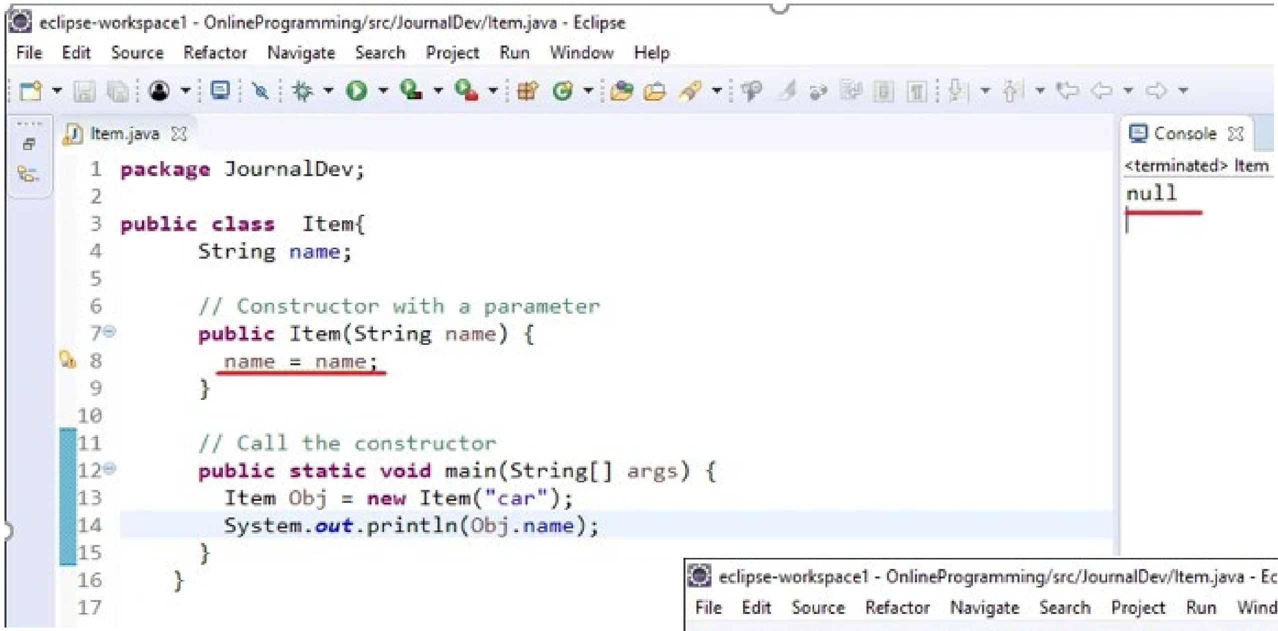


```
ArrayList<Object> a = new ArrayList<Object>();
add(Object o);
remove(Object o);
```

(6) Others

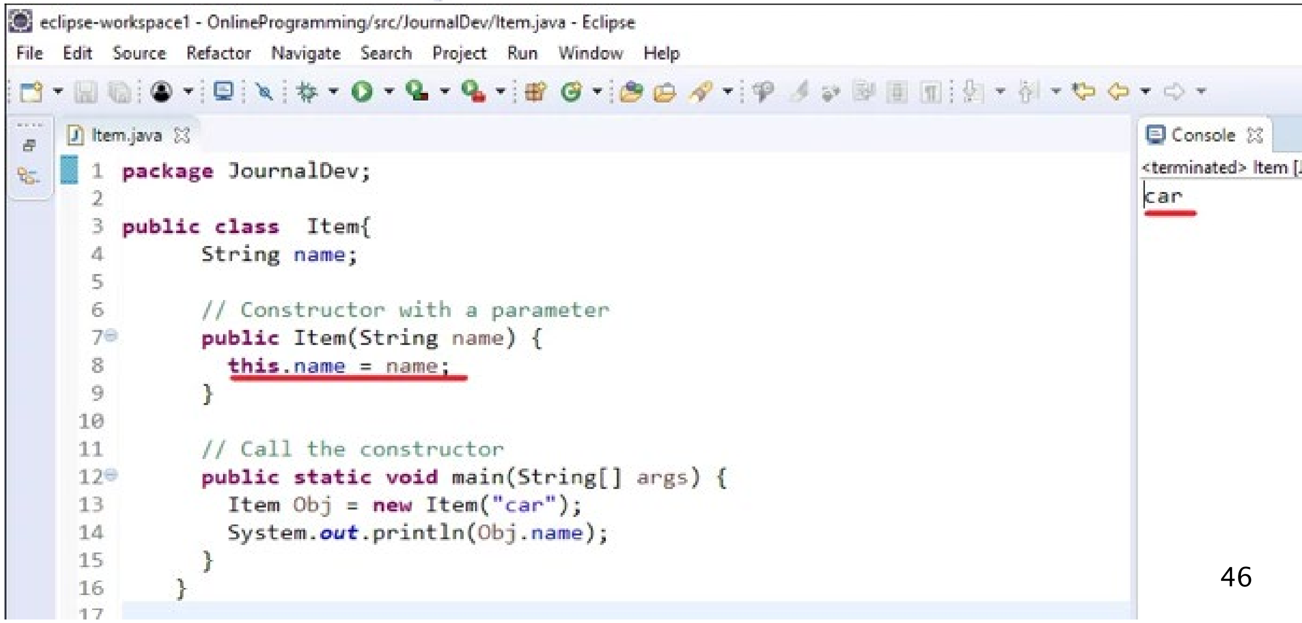
- Keyword "this"
  - **this keyword in Java** is a reference variable that refers to the current object of a method or a constructor.
  - The main purpose of using this keyword in Java is to remove the confusion between class attributes and parameters that have same names.

(6) Others


eclipse-workspace1 - OnlineProgramming/src/JournalDev/Item.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Item.java
Console
<terminated> Item

```
package JournalDev;

public class  Item{
      String name;

      // Constructor with a parameter
      public Item(String name) {
        name = name;
      }

      // Call the constructor
      public static void main(String[] args) {
        Item Obj = new Item("car");
        System.out.println(Obj.name);
      }
    }
```

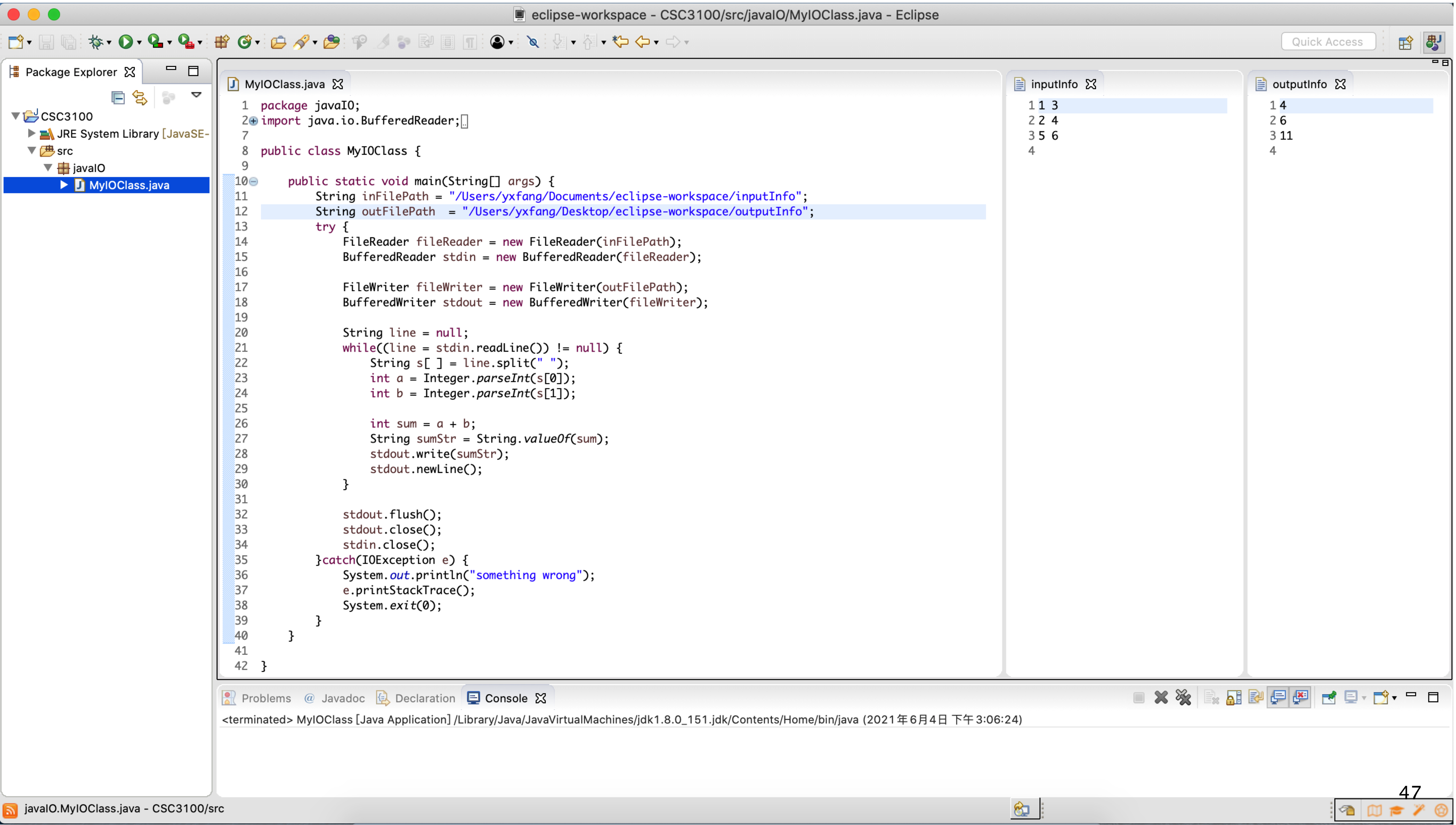
```
null
```


eclipse-workspace1 - OnlineProgramming/src/JournalDev/Item.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Item.java
Console
<terminated> Item [

```
package JournalDev;

public class  Item{
      String name;

      // Constructor with a parameter
      public Item(String name) {
        this.name = name;
      }

      // Call the constructor
      public static void main(String[] args) {
        Item Obj = new Item("car");
        System.out.println(Obj.name);
      }
    }
```

```
car
```

(6) Others



eclipse-workspace - CSC3100/src/javaIO/MyIOClass.java - Eclipse

MyIOClass.java

```java
package javaIO;
import java.io.BufferedReader;

public class MyIOClass {

    public static void main(String[] args) {
        String inFilePath = "/Users/yxfang/Documents/eclipse-workspace/inputInfo";
        String outFilePath  = "/Users/yxfang/Desktop/eclipse-workspace/outputInfo";
        try {
            FileReader fileReader = new FileReader(inFilePath);
            BufferedReader stdin = new BufferedReader(fileReader);

            FileWriter fileWriter = new FileWriter(outFilePath);
            BufferedWriter stdout = new BufferedWriter(fileWriter);

            String line = null;
            while((line = stdin.readLine()) != null) {
                String s[ ] = line.split(" ");
                int a = Integer.parseInt(s[0]);
                int b = Integer.parseInt(s[1]);

                int sum = a + b;
                String sumStr = String.valueOf(sum);
                stdout.write(sumStr);
                stdout.newLine();
            }

            stdout.flush();
            stdout.close();
            stdin.close();
        }catch(IOException e) {
            System.out.println("something wrong");
            e.printStackTrace();
            System.exit(0);
        }
    }

}
```

inputInfo

```
1 3
2 4
5 6

```

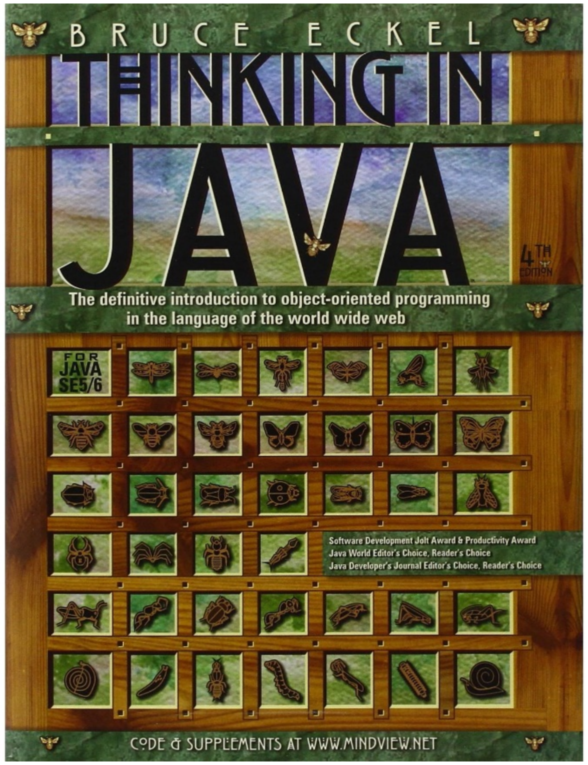
outputInfo

```
4
6
11

```

```
<terminated> MyIOClass [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_151.jdk/Contents/Home/bin/java (2021年6月4日 下午3:06:24)
```

More inforamtion

- More online materials
  - https://www.guru99.com/java-tutorial.html
  - https://fileadmin.cs.lth.se/cs/Education/EDA040/common/java21.pdf
  - Book: Think in Java


BRUCE ECKEL
THINKING IN JAVA
4TH EDITION
The definitive introduction to object-oriented programming in the language of the world wide web
FOR JAVA SE5/6
Software Development Jolt Award & Productivity Award
Java World Editor's Choice, Reader's Choice
Java Developer's Journal Editor's Choice, Reader's Choice
CODE & SUPPLEMENTS AT WWW.MINDVIEW.NET