

101: Java

May, 2023



Objective

- インストール: Eclipse
- Hello world!
- 文法
- Data types
- Operators (演算子)
- Loops (繰り返し処理)
- Condition (条件分岐)
- Array (配列)
- Exception (例外)
- Files and I/O
- Scanner

Eclipse IDE

- Eclipse IDE
 - <https://www.eclipse.org/downloads/>

The Eclipse Installer 2023-03 R now includes a JRE for macOS, Windows and Linux.



Get **Eclipse IDE 2023-03**

Install your favorite desktop IDE packages.

[Download x86_64](#)

[Download Packages](#) | [Need Help?](#)

eclipseinstaller by Oomph

type filter text



Eclipse IDE for Java Developers

The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle integration



Eclipse IDE for Enterprise Java and Web Developers

Tools for developers working with Java and Web applications, including a Java IDE, tools for JavaScript, TypeScript, JavaServer...



Eclipse IDE for C/C++ Developers

An IDE for C/C++ developers.



Eclipse IDE for Embedded C/C++ Developers

An IDE for Embedded C/C++ developers. It includes managed cross build plug-ins (Arm and RISC-V) and debug plug-ins (SEGGER...

MJBK防止策: UTF-8

For example, some Shift-JIS characters include a [backslash](#) (0x5C "\") in the second byte, which is used as an [escape character](#) in many programming languages.

構		わ		な		い	
8d	5c	82	ed	82	c8	82	a2

A parser lacking support for Shift JIS will recognize 0x5C 0x82 as an invalid escape sequence, and remove it.^[3] Therefore, the phrase cause mojibake.

高		塀		ね		い	
8d		82	ed	82	c8	82	a2

Hello world

```
public class MyFirstJavaProgram {                                //class

    /* First java program.
     * This will print 'Hello world' as the output
     */

    public static void main(String[] args) {                    //main() method not Main
        System.out.println("Hello world");                      /*prints "Hello world"*/
    }
}
```

```
javac MyFirstJavaProgram.java  
java MyFirstJavaProgram
```

```
#include <stdio.h>  
int main() {  
    printf("Hello, world!\n");  
    return 0;  
}
```



```
#include <iostream>  
int main() {  
    std::cout << "Hello world!\n";  
    return 0;  
}
```



```
print("Hello world")
```



Object-oriented programming (OOP)

- Objects



States(属性/状態)

♂/♀、年齢、白柴/胡麻柴/黒柴, etc.

Behaviors(操作/動作)

- Class

- blueprint/template (e.g. DNA, RNA)



例: class Dog

```
public class Dog {  
    String breed;  
    int age;  
    String color;  
  
    void bark() {  
        System.out.println("Wan wan");  
    }  
  
    void eat() {  
    }  
  
    void sleep() {  
    }  
}
```

Keywords: abstract, boolean, break, byte, case, catch, char, class, do, for, if, ...

Constructors

MyClass.java

```
public class MyClass {  
    int num;  
    MyClass() {  
        num = 100;  
    }  
}
```

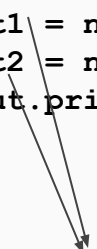
constructor



ConsDemo.java


```
public class ConsDemo {  
    public static void main(String[] args) {  
        MyClass t1 = new MyClass();  
        MyClass t2 = new MyClass();  
        System.out.println(t1.num + " " + t2.num);  
    }  
}
```


new objects

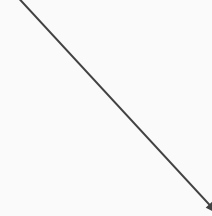


Constructors

MyClass2.java

```
class MyClass2 {  
    int x;  instance variable  
  
    MyClass2(int i) {  
        x = i;  
    }  
}
```

 constructor

 parameter

ConsDemo2.java

```
public class ConsDemo2 {  
    public static void main(String[] args) {  
        MyClass2 t1 = new MyClass2(10);  
        MyClass2 t2 = new MyClass2(20);  
        System.out.println(t1.x + " " + t2.x);  
    }  
}
```

- a special method, matches the class name
- does not have a return type (void, int, etc.)
- is called when the object is created
- All Java classes have constructors

```
public class Puppy {  
    int puppyAge;                                //instance variable, outside any method  
  
    Puppy(String name) {  
        // This constructor* has one parameter, name.  
        System.out.println("The name is: " + name);  
    }  
  
    public void setAge(int age) {                //This method takes one parameter, age.  
        puppyAge = age;  
    }  
  
    public int getAge() {  
        System.out.println("Puppy's age is: " + puppyAge);  
        return puppyAge;  
    }  
  
    public static void main(String[] args) {  
        // create an object myPuppy  
        Puppy myPuppy = new Puppy("Cody");  
  
        myPuppy.setAge(2);  
  
        myPuppy.getAge();  
  
        /* read age again */  
        System.out.println("Variable Value: " + myPuppy.puppyAge);  
    }  
}
```

Primitive data types

- **byte** (8 bit signed integer)
 - `byte a = 100;`
 - min: -128 (-2^7), max: 127 (2^7-1)
- **short** (16 bit signed integer)
 - `short b = 11000;`
 - min: -32,768 (-2^{15}), max: 32,767 ($2^{15}-1$)
- **int** (32 bit signed integer)
 - `int c = -2000000;`
 - min: -2^{31} , max: $2^{31}-1$
- **long** (64 bit signed integer)
 - min: -2^{63} , max: $2^{63}-1$
- **float** (32 bit single precision floating point)
 - `float f = 22.22;`
- **double** (64 bit double precision floating point)
 - `double d = 33.4444;`
- **boolean** default: false
 - `boolean isAlive = true;`
 - `boolean isDead = false;`
- **char** (single 16 bit Unicode character)
 - `char letterA = 'A';`

default: 0

Reference data types

```
Puppy myPuppy = new Puppy("Cody");
```



reference variable

default: null

Variable types

```
public class Test {  
    public void puppyAge() {  
        int age = 0;  
        age = age + 7;  
        System.out.println("Puppy age is: " + age);  
    }  
  
    public static void main(String[] args) {  
        Test test = new Test();  
        test.puppyAge();  
    }  
}
```

```
public class Test {  
    public void puppyAge() {  
        int age;  
        age = age + 7;  
        System.out.println("Puppy age is: " + age);  
    }  
  
    public static void main(String[] args) {  
        Test test = new Test();  
        test.puppyAge();  
    }  
}
```

//エラー、初期化されていない

Local variables: declared in methods, constructors

“変数宣言”

no modifiers, no default value



```
public class Employee {  
  
    // this instance variable is visible for any child class.  
    public String name;  
  
    // salary variable is visible in Employee class only.  
    private double salary;  
  
    // The name variable is assigned in the constructor.  
    public Employee (String empName) {  
        name = empName;  
    }  
  
    // The salary variable is assigned a value.  
    public void setSalary(double empSal) {  
        salary = empSal;  
    }  
  
    // This method prints the employee details.  
    public void printEmp() {  
        System.out.println("name: " + name);  
        System.out.println("salary:" + salary);  
    }  
  
    public static void main(String[] args) {  
        Employee empOne = new Employee("Rajesh");  
        empOne.setSalary(100000);  
        empOne.printEmp();  
    }  
}
```

instance variable

declared outside any method or constructor, inside a class
"public""private": modifiers
in a static method, should be called using fully qualified name.
empOne.name

no nested method



```
public class Employee {
```

```
    // salary variable is a private static variable
```

```
    private static double salary;
```

```
    // DEPARTMENT is a constant
```

```
    public static final String DEPARTMENT = "システム開発部";
```

```
    public static void main(String[] args) {
```

```
        salary = 300000;
```

```
        System.out.println(DEPARTMENT + " average salary: " + salary);
```

```
    }
```

```
}
```

class/static variables

from outside class:
Employee.DEPARTMENT

Modifiers

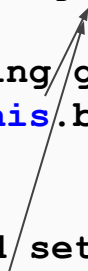
- Access modifiers

- (空欄)
- **public**
- **private**
- **protected**

- Non access modifiers

- **static** (method, class)
- **final**
- **abstract**
- **synchronized***, **volatile***

```
public class Dog {  
    private String birthday;  
  
    public String getBirthday() {  
        return this.birthday;  
    }  
  
    public void setBirthday(String birthday) {  
        this.birthday = birthday;  
    }  
}
```



Operators

- +
- -
- *
- /
- %
- ++
- --

- ==
- !=
- >
- <
- >=
- <=
- &&
- ||
- !
- =

Loops

```
public class TestLoop1 {  
  
    public static void main(String[] args) {  
        int x = 10;  
  
        while(x < 20) {                // also try x <= 20  
            System.out.print("value of x: " + x);  
            x++;  
            System.out.print("\n");    //手動で改行 escape sequence  
        }  
    }  
}
```

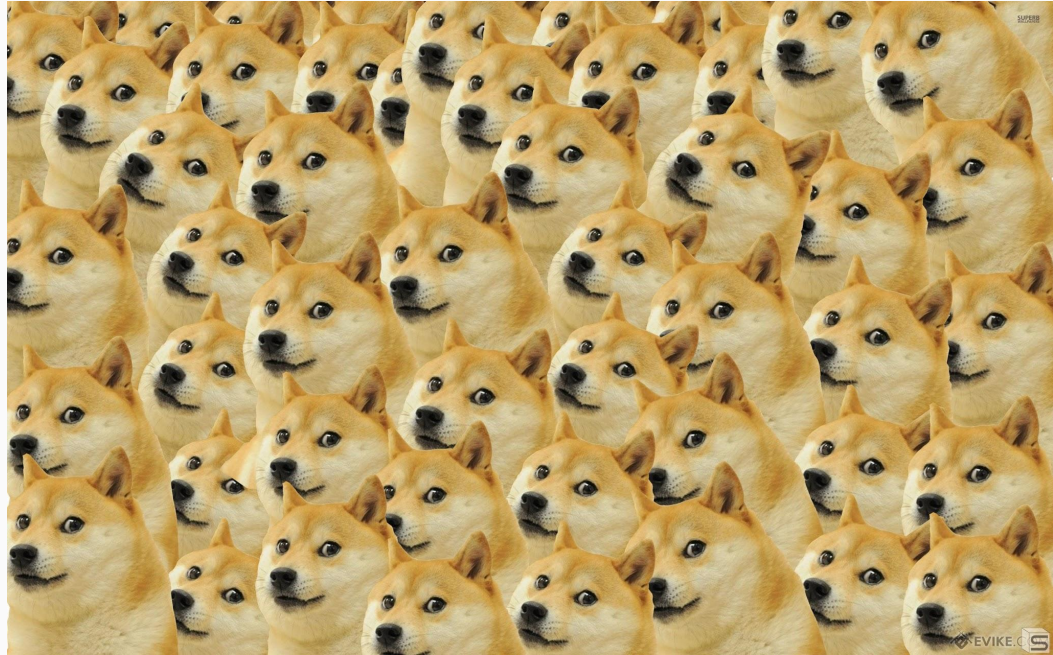
Loops

```
public class TestLoop2 {  
  
    public static void main(String[] args) {  
  
        for(int x = 10; x < 20; x = x + 1) {  
            System.out.println("value of x: " + x );  
        }  
    }  
}
```

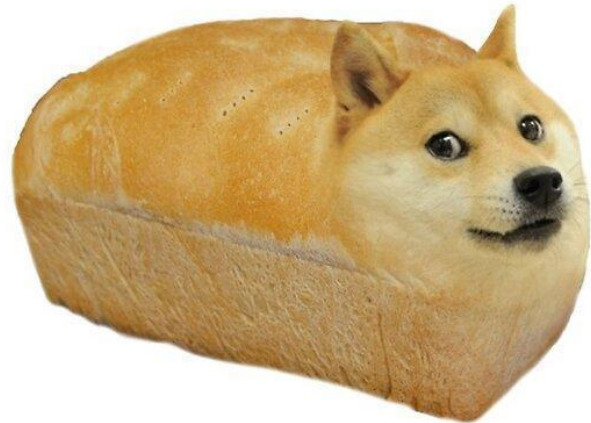
Loops

```
public class TestLoop3 {  
  
    public static void main(String[] args) {  
        int x = 10;  
  
        do {  
            System.out.println("value of x: " + x );  
            x++;  
        } while(x < 20);  
    }  
}
```

```
public class TestLoop4 {  
  
    public static void main(String[] args) {  
        int i = 8;  
        while (true) {  
            System.out.println(i);  
            i++;  
            if (i == 20) {  
                System.out.  
println("i is 20");  
                break;  
            }  
        }  
    }  
}
```



```
public class TestLoop5 {  
  
    public static void main(String[] args) {  
        int x = 8;  
        for (int i=1; i<10; i++) {  
            x++;  
            if (x % 3 == 0) {  
                continue;  
            }  
            System.out.println(x + " x%3!=0");  
        }  
        System.out.println("Value of x: " + x);  
    }  
}
```




```
public class TestLoop6 {  
  
    public static void main(String[] args) {  
        int x = 8;  
        for (int i=1; i<10; i++) {  
            num++;  
            if (x % 2 == 0) {  
                System.out.println("even number: " + x);  
            } else {  
                System.out.println("odd number: " + x);  
            }  
        }  
    }  
}
```

```
public class TestLoop7 {  
  
    public static void main(String[] args) {  
        int time = 0;  
        for (int i=1; i<=24; i++) {  
            if (time < 12) {  
                System.out.println("Morning: " + time);  
            } else if (time <= 18) {  
                System.out.println("Afternoon: " + time);  
            } else {  
                System.out.println("Evening: " + time);  
            }  
            time++;  
        }  
    }  
}
```

```
public class TestLoop8 {  
  
    public static void main(String[] args) {  
        int x = 1, y = 2;  
  
        for (int i=1; i<11; i++) {  
            if (x < 15) {  
                x++;  
                if (x != y) {  
                    y = y + 2;  
                }  
            }  
            System.out.println("x: " + x + "\ny: " + y + "\n");  
        }  
    }  
}
```

Exercise: nested for loop

```
*****  
*****  
*****  
*****  
*****
```

5*8

```
for (...) {  
    for (...) {  
        System.out.print("*") ;  
    }  
    ...  
}
```

```
public class TestSwitch {  
    public static void main(String[] args) {  
        char grade = 'C';           // 任意の値を  
  
        switch(grade) {  
            case 'A':  
                System.out.println("Excellent!");  
                break;  
            case 'B':  
            case 'C':  
                System.out.println("Well done");  
                break;  
            case 'D':  
                System.out.println("You passed");  
            case 'F':  
                System.out.println("Better try again");  
                break;  
            default:  
                System.out.println("Invalid grade");  
        }  
        System.out.println("Your grade is " + grade);  
    }  
}
```

Data type conversion (型変換)

```
public void method1() {  
    byte a = 18;  
    int b = 157;  
    long c = 5000000L;  
    float f = 3.14f;  
    double d = 88.8888888888;  
    char j = 'j';  
    ...  
}
```

Unicode表

j: 0x006A (16進数), 106 (10進数)

```
int ab = a + b;           //OK  
byte ba = a + b; //NG, compile error  
float bf = b + f;         //OK  
float cf = c + f;         //OK  
double bcfd = b+c+f+d;    //OK  
int da = d + a;           //NG  
  
int f2 = (int)f;          // 3  
  
int j1 = j + 1;           // 107
```

String (文字列)

```
String str = Integer.toString(b);  
String str2 = String.valueOf(c);
```

```
System.out.println(Integer.parseInt("-20"));           // -20
```

```
int hoge = Integer.valueOf("12345");                   // valueOf() returns Integer  
int fuga = Integer.parseInt("12345");                   // parseInt() returns int
```

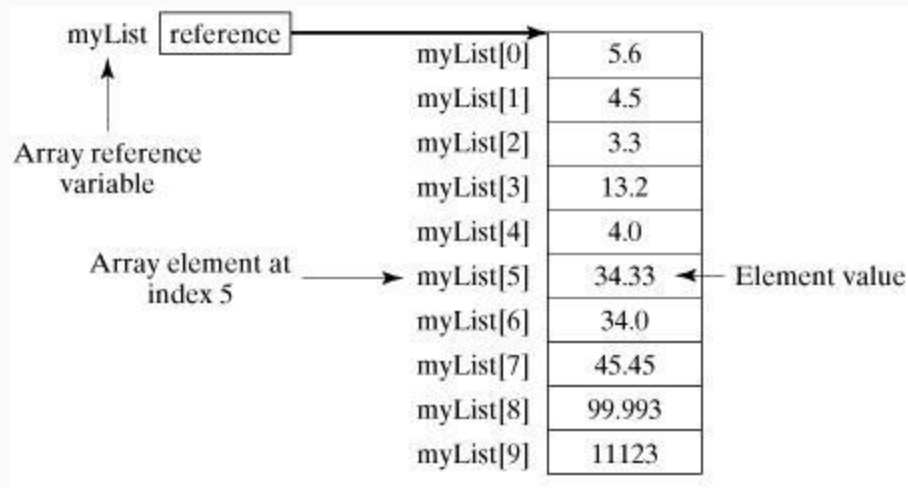
Arrays (配列)

```
String[] args
```

```
double[] myList  
= {5.6, 4.5, 3.3, 13.2,  
4.0, 34.33, 34.0, 45.45,  
99.993, 11123};
```

```
int[] myList2 = new int[5];
```

固定



Print the array: TestArray1.java

```
public class TestArray1 {  
    public static void main(String[] args) {  
        double[] myList = {5.6, 4.5, 3.3, 13.2, 4.0, 34.33, 34.0, 45.45,  
99.993, 11123};  
        System.out.println(myList);  
    }  
}
```

```
$ javac TestArray1.java
```


```
$ java TestArray1
```

```
[D@7ad041f3
```



reference variable

MEMORY



0X1000	0X4
0X1004	0X1000
0X1008	
0X100C	
0X1010	
0X1014	
0X1018	

ADDRESS

VALUE

```
int x = 4;  
int *pX = &x;
```

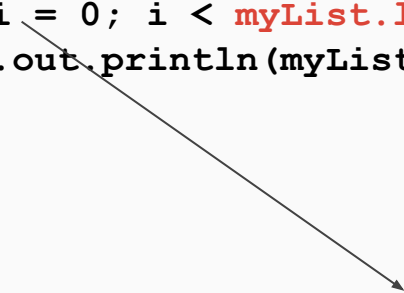
Exercise 1. print all array elements
3. find max

2. sum all elements

```
public class TestArray2 {  
  
    public static void main(String[] args) {  
        double[] myList = {5.6, 4.5, 3.3, 13.2, 4.0, 34.33, 34.0, 45.45,  
99.993, 11123};  
  
        // Print all the array elements  
        for (int i = 0; i < myList.length; i++) {  
            System.out.println(myList[i] + " ");  
        }  
        for () {}  
        for () {}  
    }  
}
```

Foreach loop: enhanced for loop

```
for (double element: myList){  
    System.out.println(element + " ");  
}
```



Sort the array: int, char, and String

```
Arrays.sort(arr);      Arrays.sort(arr, 1, 5);  
Arrays.sort(arr, Collections.reverseOrder());
```



Exercise: sort an int array without using
.sort()

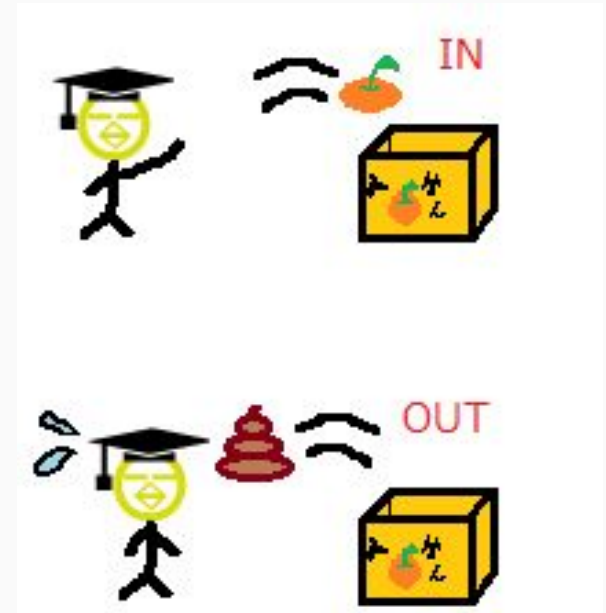
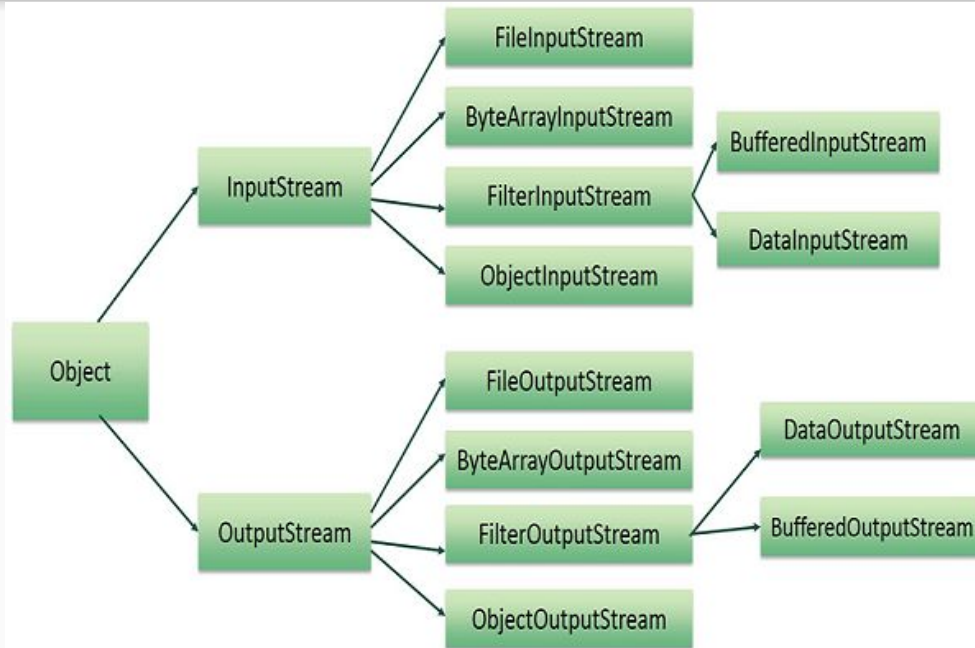
Exceptions

```
jing@jing-ThinkPad-E490:~/Documents$ nano TestArray1.java
jing@jing-ThinkPad-E490:~/Documents$ javac TestArray1.java
jing@jing-ThinkPad-E490:~/Documents$ java TestArray1
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 100 out of bounds for length 6
    at TestArray1.main(TestArray1.java:4)
jing@jing-ThinkPad-E490:~/Documents$
```

```
jing@jing-ThinkPad-E490:~/Documents$ javac TestArray1.java
jing@jing-ThinkPad-E490:~/Documents$ java TestArray1
Exception in thread "main" java.lang.NullPointerException: Cannot load from double array because "<local1>" is null
    at TestArray1.main(TestArray1.java:5)
jing@jing-ThinkPad-E490:~/Documents$
```

More on that later

Files and I/O



CopyFile1.java

```
import java.io.*;

public class CopyFile1 {

    public static void main(String[] args) throws IOException {
        FileInputStream in = null;
        FileOutputStream out = null;           //初期化

        try {
            in = new FileInputStream("input.txt");
            out = new FileOutputStream("output.txt");

            int c;
            while ((c = in.read()) != -1) {
                out.write(c);
            }
        } finally {
            if (in != null) {
                in.close();
            }
            if (out != null) {
                out.close();
            }
        }
    }
}
```

input.txt

Tesuto

byte stream
1 byte (8 bit)

①

```
InputStream in = new  
FileInputStream("C:/java/input.txt  
");
```

②

```
File f = new File("~/input.txt");  
InputStream in = new  
FileInputStream(f);
```

end of file

CopyFile2.java

input.txt

Tesuto

```
import java.io.*;
public class CopyFile2 {
```

```
    public static void main(String[] args) throws IOException {
```

```
        FileReader in = null;
```

```
        FileWriter out = null;
```

```
        try {
```

```
            in = new FileReader("input.txt");
```

```
            out = new FileWriter("output.txt");
```

```
            int c;
```

```
            while ((c = in.read()) != -1) {
```

```
                out.write(c);
```

```
            }
```

```
        }finally {
```

```
            if (in != null) {
```

```
                in.close();
```

```
            }
```

```
            if (out != null) {
```

```
                out.close();
```


```
            }
```

```
        }
```

```
    }
```

```
}
```

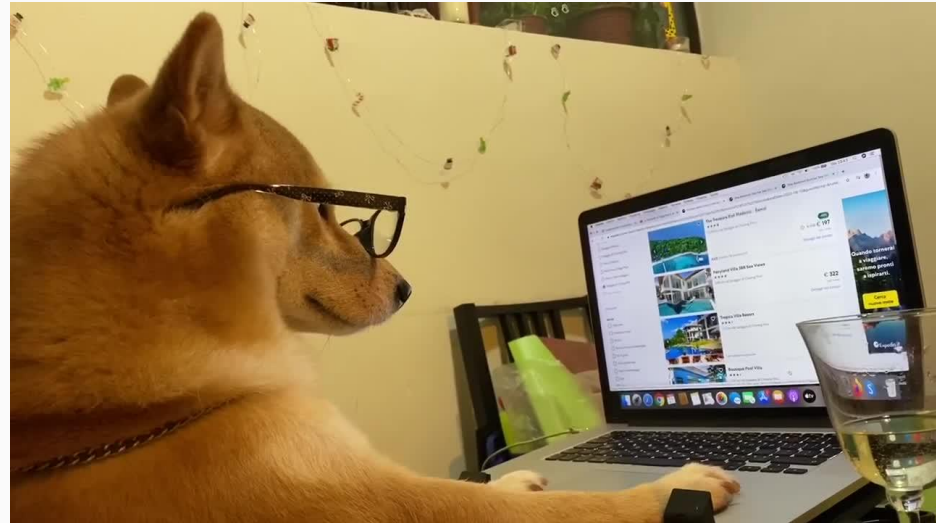
character stream
2 bytes (16 bit)




```
import java.io.*;
public class ReadConsole {

    public static void main(String[] args) throws IOException {
        InputStreamReader cin = null;

        try {
            cin = new InputStreamReader(System.in);
            System.out.println("Enter characters, 'q' to quit.");
            char c;
            do {
                c = (char) cin.read();
                System.out.print(c);
            } while(c != 'q');
        } finally {
            if (cin != null) {
                cin.close();
            }
        }
    }
}
```



Scanner

```
import java.util.Scanner;
public class ScannerDemo1 {
    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);

        String name = sc.nextLine();           // String input
        int age = sc.nextInt();

        System.out.println("Name: "+name);
        System.out.println("Age: "+age);
    }
}
```

Exercise: calculate the mean

Scannerクラスを使って、入力された整数値の平均値を求めなさい。

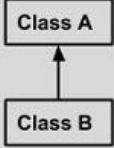
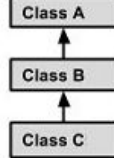
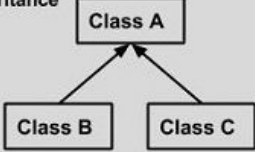
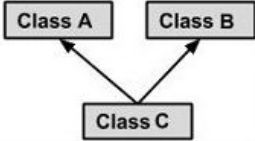
Hint: `boolean Scanner.hasNextInt()`

`boolean Scanner.hasNextLine()`

`void Scanner.close()`

Inheritance: **extends**

- the subclass (child class) inherits methods and fields from the superclass (parent class)

Single Inheritance	 <pre>graph BT; B[Class B] --> A[Class A]</pre>	<pre>public class A { } public class B extends A { }</pre>
Multi Level Inheritance	 <pre>graph BT; C[Class C] --> B[Class B]; B --> A[Class A]</pre>	<pre>public class A { } public class B extends A { } public class C extends B { }</pre>
Hierarchical Inheritance	 <pre>graph BT; B[Class B] --> A[Class A]; C[Class C] --> A</pre>	<pre>public class A { } public class B extends A { } public class C extends A { }</pre>
Multiple Inheritance	 <pre>graph BT; A[Class A] --> C[Class C]; B[Class B] --> C</pre>	<pre>public class A { } public class B { } public class C extends A, B { } // Java does not support multiple Inheritance</pre>



```
class Calculation {

    public int addition(int x, int y) {
        return x + y;
    }

    public int subtraction(int x, int y) {
        return x - y;
    }
}

public class My_Calculation extends Calculation {
    public int multiplication(int x, int y) {
        return x * y;
    }

    public static void main(String[] args) {
        int a = 20, b = 10;
        My_Calculation demo = new My_Calculation();
        System.out.println(demo.addition(a, b));
        System.out.println(demo.subtraction(a, b));
        System.out.println(demo.multiplication(a, b));
    }
}
```

```
class Super_class {
    int num = 20;

    public void display() { System.out.println("This is the superclass"); }
}

public class Sub_class extends Super_class {
    int num = 10;

    public void display() {System.out.println("This is the subclass"); }

    public void my_method() {
        sub.display();
        super.display();

        System.out.println("sub class:"+ sub.num);
        System.out.println("super class:"+ super.num);
    }

    public static void main(String[] args) {
        Sub_class obj = new Sub_class();
        obj.my_method();
    }
}
```

```
class Animal {  
}  
  
class Mammal extends Animal {  
}  
  
class Reptile extends Animal {  
}  
  
public class Dog extends Mammal {  
  
    public static void main(String[] args) {  
        Animal a = new Animal();  
        Mammal m = new Mammal();  
        Dog d = new Dog();  
  
        System.out.println(m instanceof Animal);  
        System.out.println(d instanceof Mammal);  
        System.out.println(d instanceof Animal);  
    }  
}
```

“Dog IS-A mammal”

Abstract

abstract methods must be overridden



```
abstract class Store {
    abstract void payment();
}

class ConvenienceStore extends Store {
    void payment() {
        System.out.println("Credit card");
    }
}

class SuperMarket extends Store {
    void payment() {
        System.out.println("Suica and Paypay");
    }
}

class FastFoodStore extends Store {
    void payment() {
        System.out.println("食券");
    }
}
```

```
public class Shopping {
    public static void main(String[] args) {
        ConvenienceStore store1 = new
        ConvenienceStore();
        SuperMarket store2 = new SuperMarket();
        FastFoodStore store3 = new
        FastFoodStore();

        store1.payment();
        store2.payment();
        store3.payment();
    }
}
```



```
abstract class Bike{
    Bike(){System.out.println("bike is created");}           // constructor
    abstract void run();
    void changeGear(){System.out.println("gear changed");}
}

class Honda extends Bike{
    void run(){System.out.println("Honda running safely..");}
}

class TestAbstraction2{
    public static void main(String[] args){
        Bike obj = new Honda();
        obj.run();
        obj.changeGear();
    }
}
```

```
/* File name : Employee.java */
public class Employee {
    private String name;
    private String idNum;          // 社員番号
    private int age;

    public String getName() {
        return name;
    }

    public String getIdNum() {
        return idNum;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int newAge) {
        age = newAge;
    }

    public void setName(String newName) {
        name = newName;
    }

    public void setIdNum(String newId) {
        idNum = newId;
    }
}
```

```
/* File name : RunEncap.java */
public class RunEncap {

    public static void main(String[] args) {
        Employee encap = new Employee();
        encap.setName("James");
        encap.setAge(20);
        encap.setIdNum("ME12343");

        System.out.print("Name: " + encap.getName() + "
Age: " + encap.getAge());
    }
}
```

Interface (is not a class)

100% abstract

```
interface ALaw {  
    public void noLiquid();  
    public void noBattery();  
}
```

```
interface BLaw {  
    public void noPets();  
}
```

```
public class AirlineA implements ALaw, BLaw {  
    public void noLiquid() {...// Ban liquids}  
    public void noBattery() {...}  
    public void noPets() {...// Ban pets}  
  
    public static void main(String[] args) {  
        // ...  
    }  
}
```

```
interface Bank{
    float rateOfInterest();
}

class UFJ implements Bank{
    public float rateOfInterest(){return 9.15f;}
}

class Mizuho implements Bank{
    public float rateOfInterest(){return 9.7f;}
}

class TestInterface2{
    public static void main(String[] args){
        Bank bank1 = new UFJ();
        Bank bank2 = new Mizuho();
        System.out.println("UFJ ROI: " + bank1.rateOfInterest());
        System.out.println("Mizuho ROI: " + bank2.rateOfInterest());
    }
}
```