Unit 1： Introduction

Data Structure

General Course Guideline

Participation is the key in this course. If you have any questions related to the lecture, don’t hesitate to ask because someone else might have the same question as you do

- **Ask**
- We assume you have some basic programming skills or knowledge in this course

Course Contents

unit 1	Introduction
unit 2	Class
unit 3	Array
unit 4	Recursion
unit 5	Complexity
unit 6	Searching and Sorting
unit 7	List
unit 8	LinkedList
unit 9	Stack
unit 10	Queue
unit 11	Tree
unit 12	Binary Search Tree
unit 13	Hash Table
unit 14	Heap

Course Tools

You can use any tool you like

- Download IDE (Integrated Development Environment)

  step1: Install JDK

  step2: Install eclipse https://www.eclipse.org

eclipse

Course Reference

a) Venugopal S. **Data Structures Outside-In with Java**[M]. Prentice-Hall, Inc., 2006.

b) Weiss M A. **Data structures and algorithm analysis in Java**[M]. Pearson Education, Inc, 2012.

c) Cormen T H, Leiserson C E, Rivest R L, et al. **Introduction to algorithms**[M]. MIT press, 2009.

d) Shaffer C A. **Data structures and algorithm analysis**[J]. 2021.

e) [1]程杰. 大话数据结构:**Play with data structure**[M]. 清华大学出版社, 2011.

Contents

1.1 Introduction to data structure

1.1

Why do We Need Data Structure

◆ Representing information

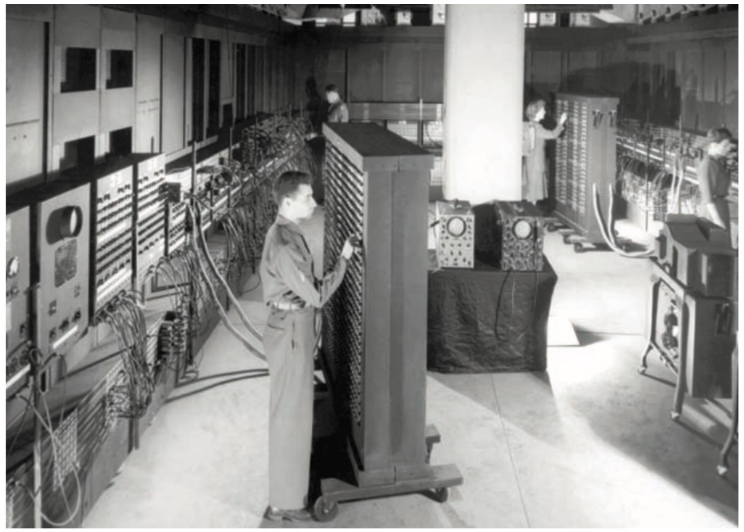
- It is not enough to have the necessary information, we must organize that information
- The primary purpose of most computer programs is not to perform calculations, but to store and retrieve information

① How many people in my company make over $100,000 per year?
② How many cities with more than 250,000 people lie within 500 miles of Dallas, Texas?
③ Find the maximum in 1000,000 numbers

Why do We Need Data Structure

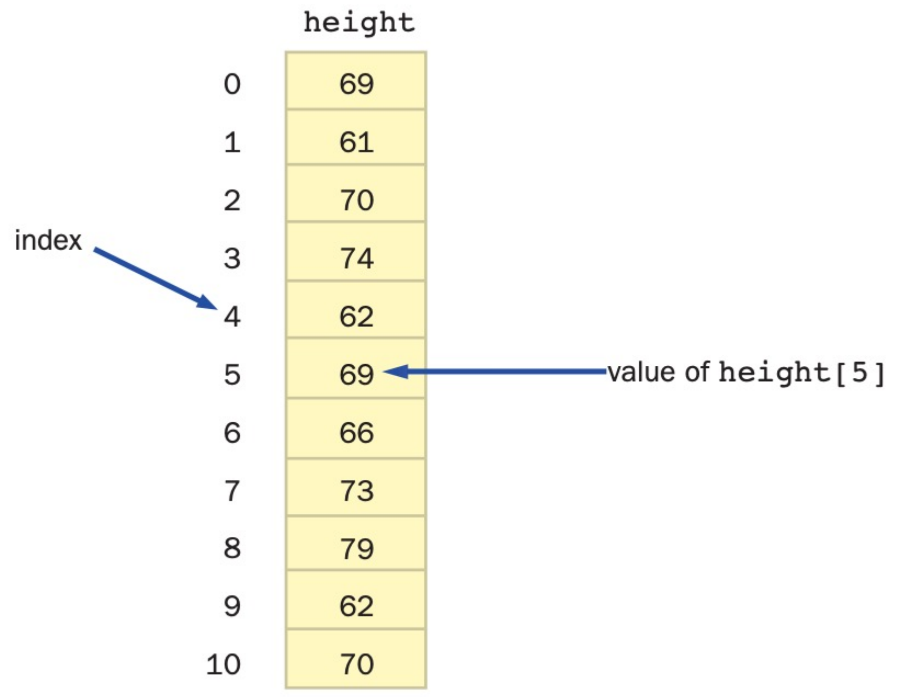
- To be fast
    - Program efficiency is always important

More complex problems demand more computation, making the need for efficient programs even greater


The ENIAC

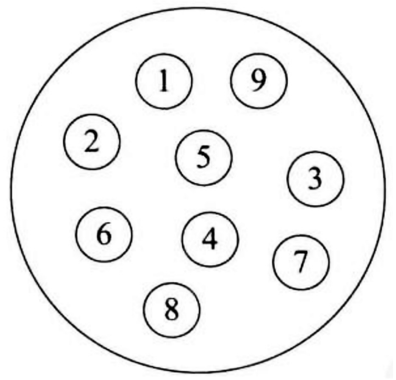
What is Data Structure

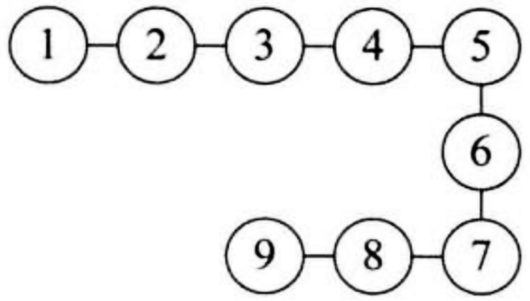
- A data structure is an organization or structuring for a collection of data items



Types of Data Structures

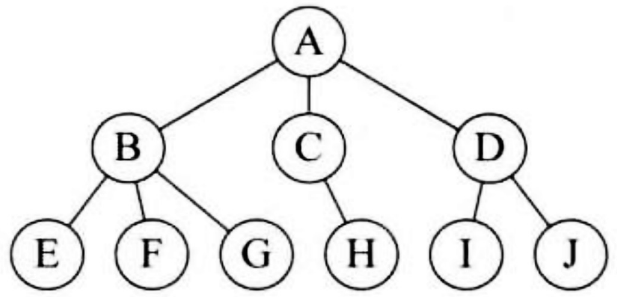
- Set 集合结构
  - Relation: Belong to the same set
- Linear structures 线性结构
  - Relation: One-to-one

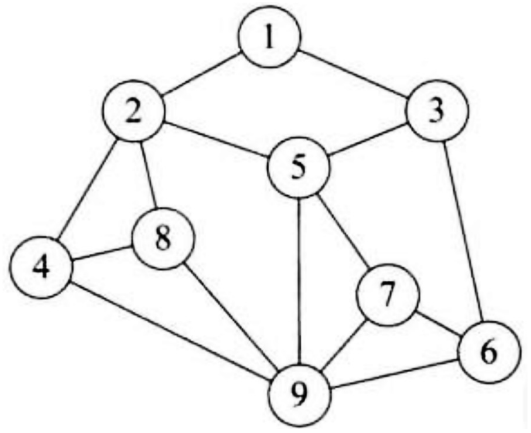




Types of Data Structures

- Trees 树形结构
  - Relation: One-to-many, hierarchy
- Graph 图形结构
  - Relation: Many-to-many





What Data Structures Do We Study

Categories	Type
Linear structures	`List`
	`LinkedList`
	`Stack`
	`Queue`
Trees	`Binary Tree`
	`Binary Search Tree`
	`AVL Tree`
	`Heap`
Others	`Hash Table`

How do We Learn Data Structure

① Step 1: Understand it

② Step 2: Implement it

We focus on the step 1 in class

I believe it is more important for a practitioner to understand the principles required to select or design the data structure that will best solve some problem than it is to memorize a lot of textbook implementations

——Shaffer C A. Data structures and algorithm analysis[J]. 2021.

Why Java for Data Structure

Java、C、C++、Python

- ◆ OOP

Java	Python
Rigorous language	Less rigorous
More code	Less code
Faster	Slower
Numerous semicolon	Nah, we don't need that stuff
Numerous parenthesis	We don't need much

Summary

1. Introduction to Data Structure

1.2 Java Basics

TESTDAILY

Introduction

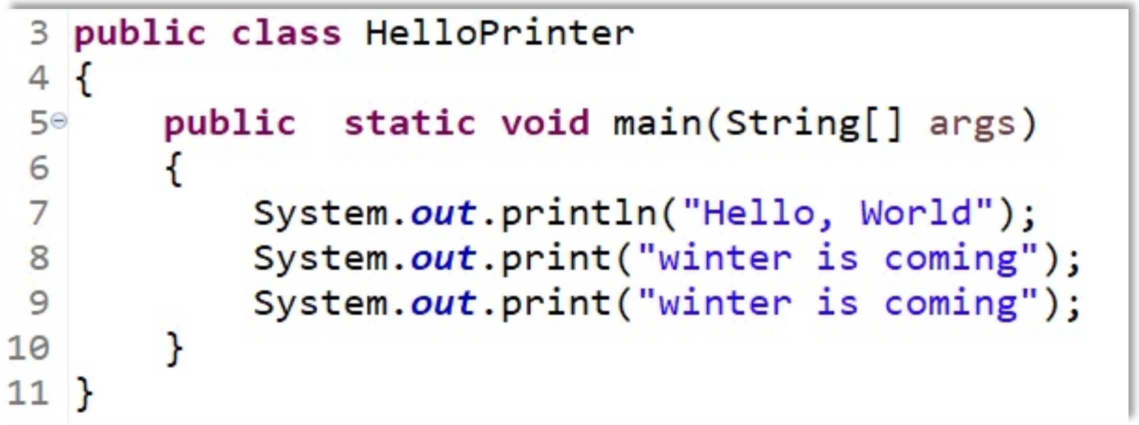
- Java Review Part I: **Java Basics** (Data types, Control structure)
- Java Review Part II: **Class** (Class, Inheritance, Interface)
- Java Review Part III: **Array** (Array, 2D Array, ArrayList)

1.2 Java Basics

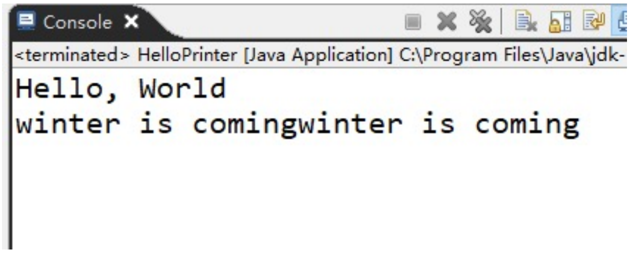
1.2.1 Data types

Output

output information on the computer monitor



```
public class HelloPrinter
{
    public  static void main(String[] args)
    {
        System.out.println("Hello, World");
        System.out.print("winter is coming");
        System.out.print("winter is coming");
    }
}
```



Console

<terminated> HelloPrinter [Java Application] C:\Program Files\Java\jdk-

```
Hello, World
winter is comingwinter is coming
```

Input

Use `Scanner` Library to help

```
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
```

```
sc.next();//input a String
sc.nextDouble()//input a double value
```

Variables 变量

- Declare and initialize a variable (the most common way)

```
<Type> <Variable name> = <some value>
```

```
int number = 10;
```

- Or by two steps:

① step 1: Declare a variable

② step 2: Initialize the variable

```
int number;
number = 10;
```

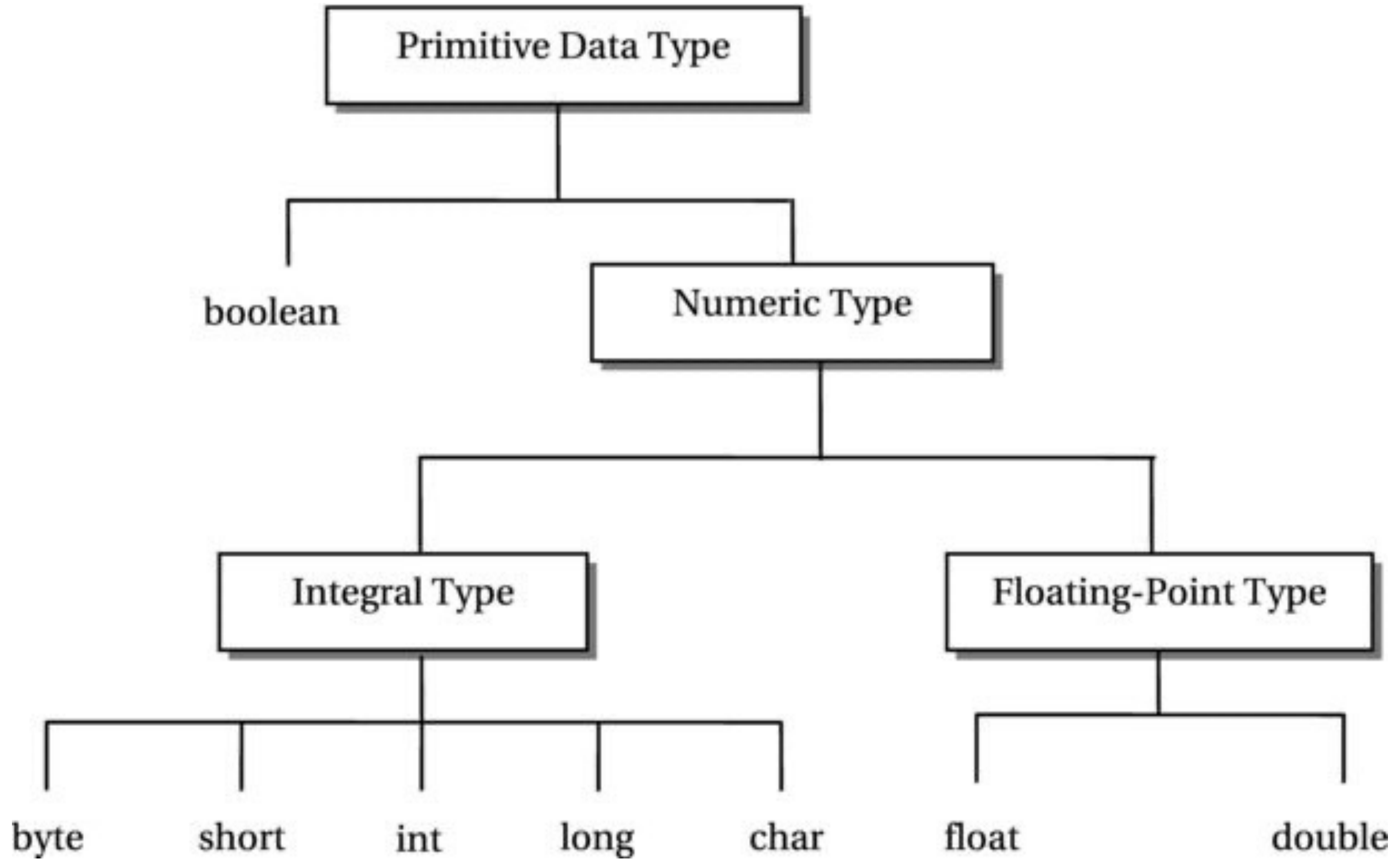
Exercises

1. What is the error in the following code segment?

```
int height;
int height = 10;
```

Concept

Primitive Types 原始类型



Primitive Types 原始类型

Categories	Type	Description
Integer	`byte`	8 bit signed integer
	`short`	16 bit signed integer
	`int`	32 bit signed integer
	`long`	64 big signed integer
Floating-point number	`float`	32 bit single-precision floating point,
	`double`	64 bit double-precision
Boolean	**`boolean`**	true/false, **1 bit information**
Character	**`char`**	single 16-bit Unicode character

Reference Types 引用类型

- Any type other than primitive type is ***reference type***

  `String:` use to store words or sentences

  ```
  String str = ”hello”;
  ```

Arithmetic Expressions

An **expression** is a combination of **operators** and **operands**, like a mathematical expression

- The arithmetic operators consist of **+, –, *, /,** and **%**

Operator	Meaning	Arithmetic Expressions
`+`	addition	`3 + x`
`–`	subtraction	`p - q`
`*`	multiplication	`6 * x`
`/`	division	`10 / 4`
`%`	reminder	`11 % 8`

Integer Division

An arithmetic operation that uses two `int` values will evaluate to an `int` value

An arithmetic operation that uses a `double` value will evaluate to a `double` value

```
10.0 / 4.0  is 2.5
10 / 4.0    is 2.5
10.0 / 4    is  2.5
10 / 4      is 2
```

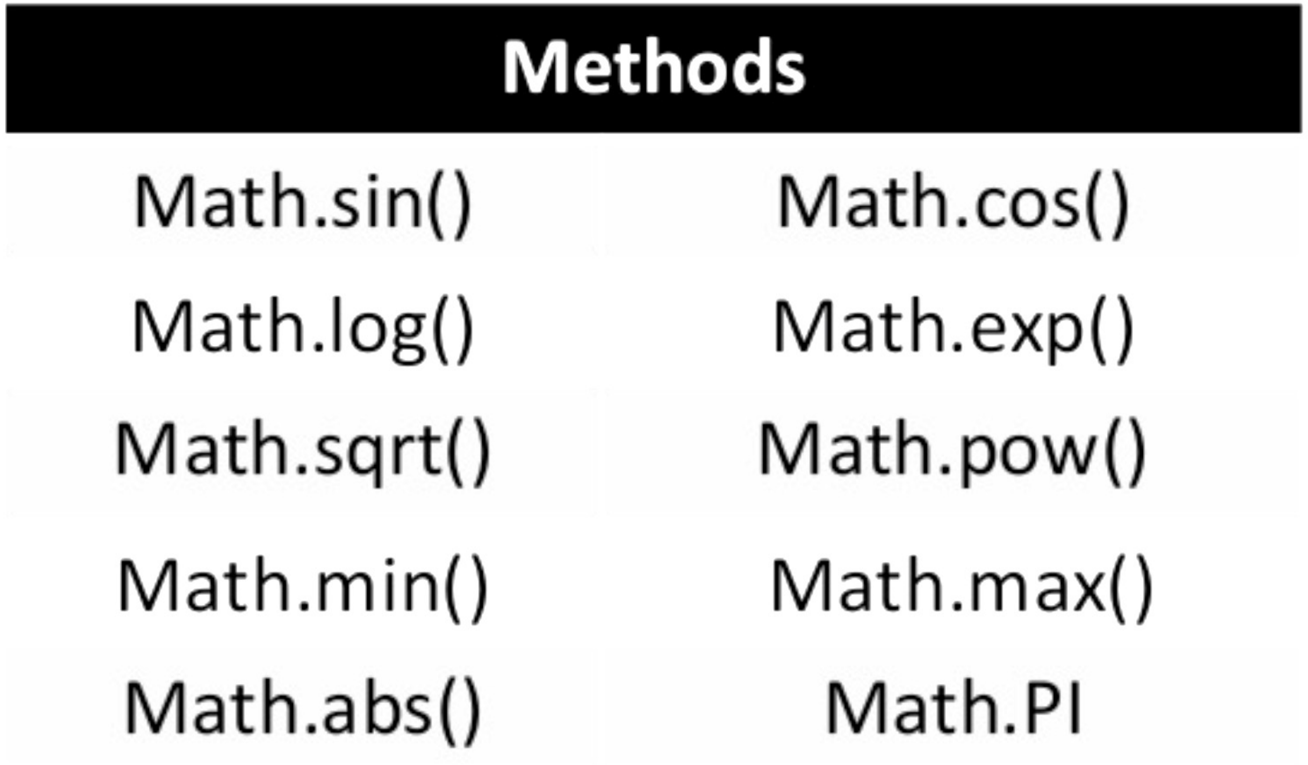
◆ An attempt to divide an integer by zero will result in an `ArithmeticException` to occur

```
System.out.println(10 / 0); //java.lang.ArithmeticException: / by zero
```

1. what is printed?

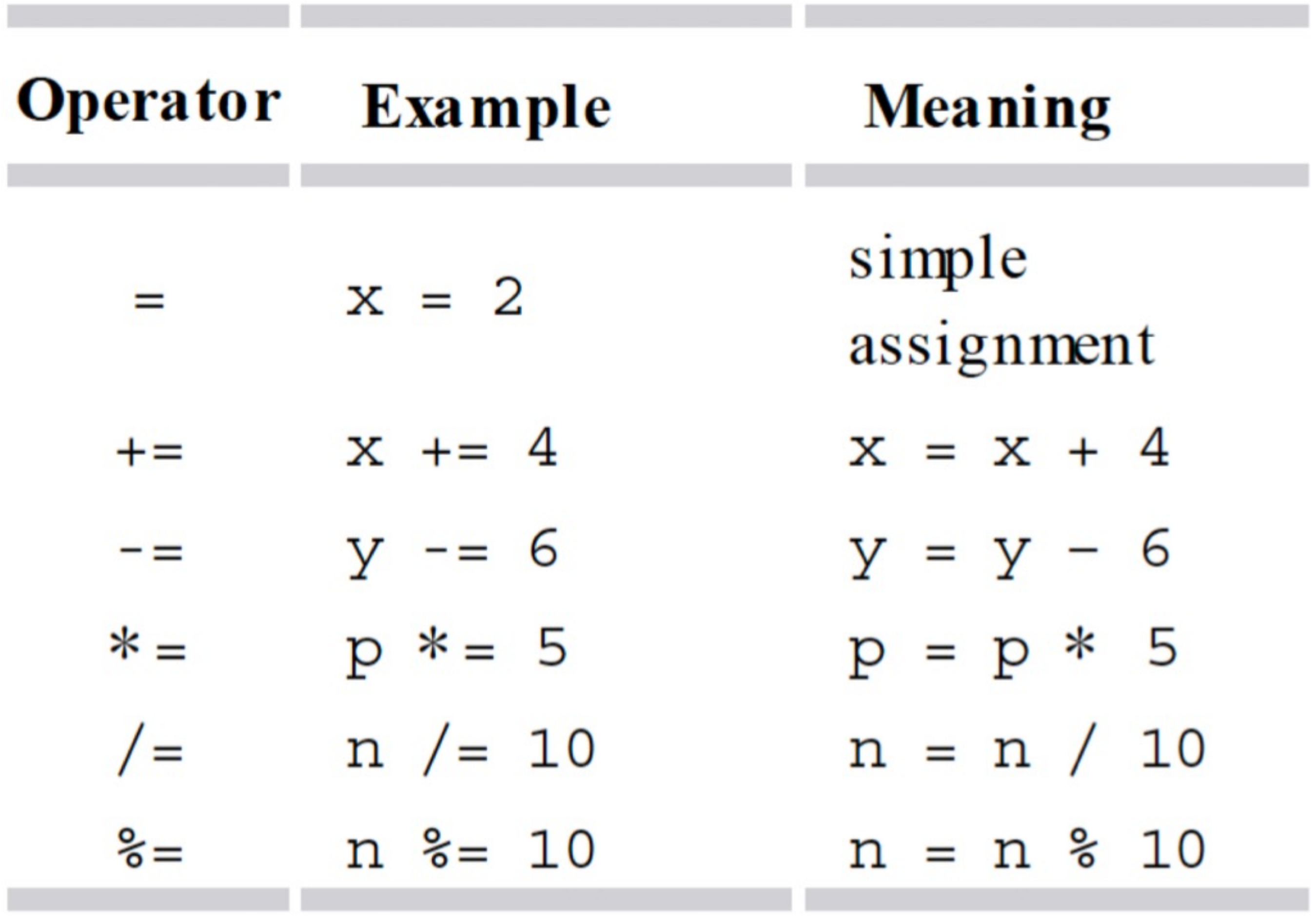
```
System.out.println(10 / 3 + 10 % 3);
```

Math library



Methods	
Math.sin()	Math.cos()
Math.log()	Math.exp()
Math.sqrt()	Math.pow()
Math.min()	Math.max()
Math.abs()	Math.PI

Assignment



Operator	Example	Meaning
=	x = 2	simple assignment
+=	x += 4	x = x + 4
-=	y -= 6	y = y - 6
*=	p *= 5	p = p * 5
/=	n /= 10	n = n / 10
%=	n %= 10	n = n % 10

Increment and Decrement Operators

- The increment operator **(++) adds 1** to any integer or floating point value

- The decrement operator **(--) subtracts 1** from the value

```
count ++;
count --;
```

⇔

```
count = count + 1;
count = count - 1;
```

`final` variable 常量

`final` variables **cannot be changed once initialized**

- In general, it is used to declare a constant that cannot be changed
- The name of a final variable is usually capitalized, such as PI, MIN_VALUE, MAX_VALUE

```
final double PI = 1.2.1415926;
final double TAX_RATE = 6.67;
PI = 9.9; // Error: The final variable PI cannot be changed
```

1. What is printed after the following code segment is executed?

```
int x = 3;
int y = 4;
x += y * 2;
x++;
System.out.println(x);
```

Concept

TESTDAILY

Variable Casting

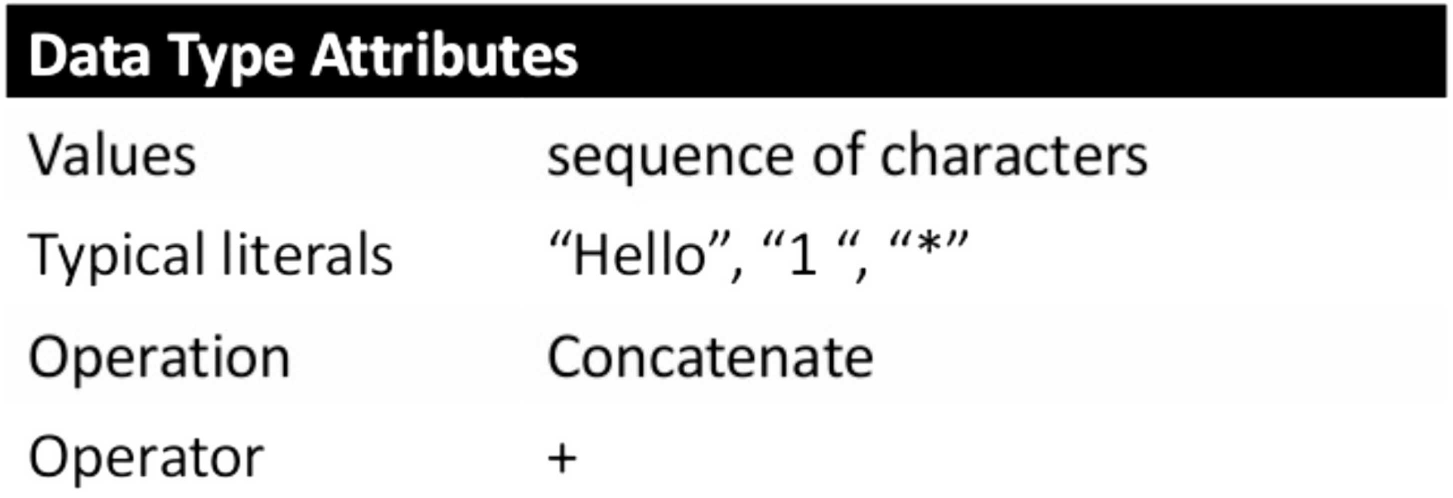
- Casting precedence

```
(int)11 * 0.3        is 1.2.1
(int)(11 * 0.3)      is 3
11 * (int)0.3        is 0

(double) 10 / 3      is 1.2.13
10 / (double) 3      is 1.2.13
(double) (10 / 3)    is 3.0
```

1. What is printed after the following code segment is executed?

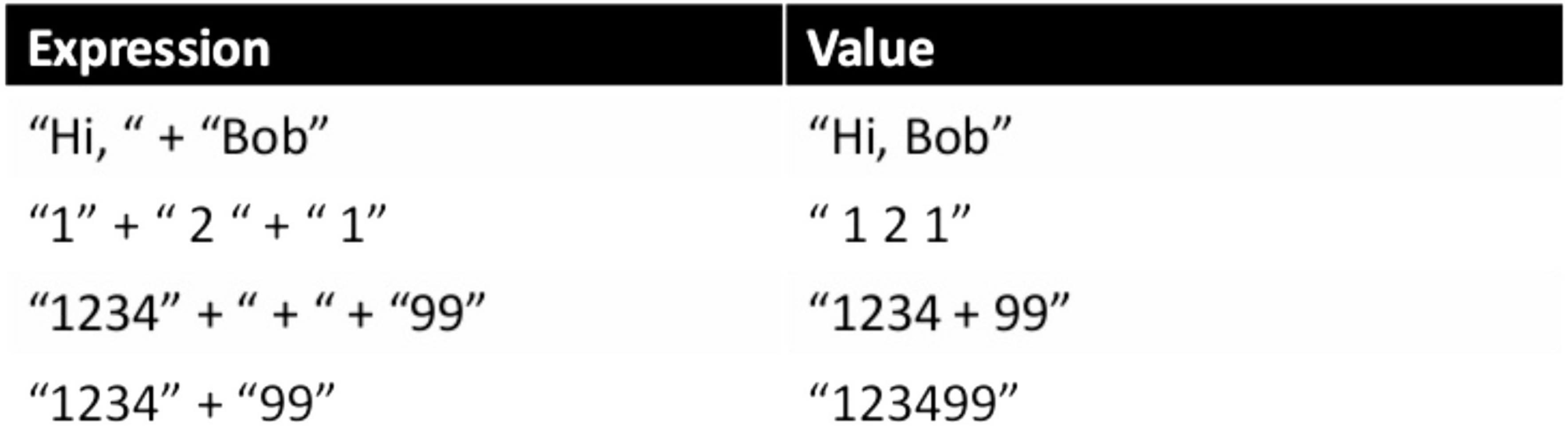
```
double x = (int)11 * 0.2 + (int)(11 * 0.2);
System.out.println(x);
```

String



Data Type Attributes	
Values	sequence of characters
Typical literals	“Hello”, “1 “, “*”
Operation	Concatenate
Operator	+

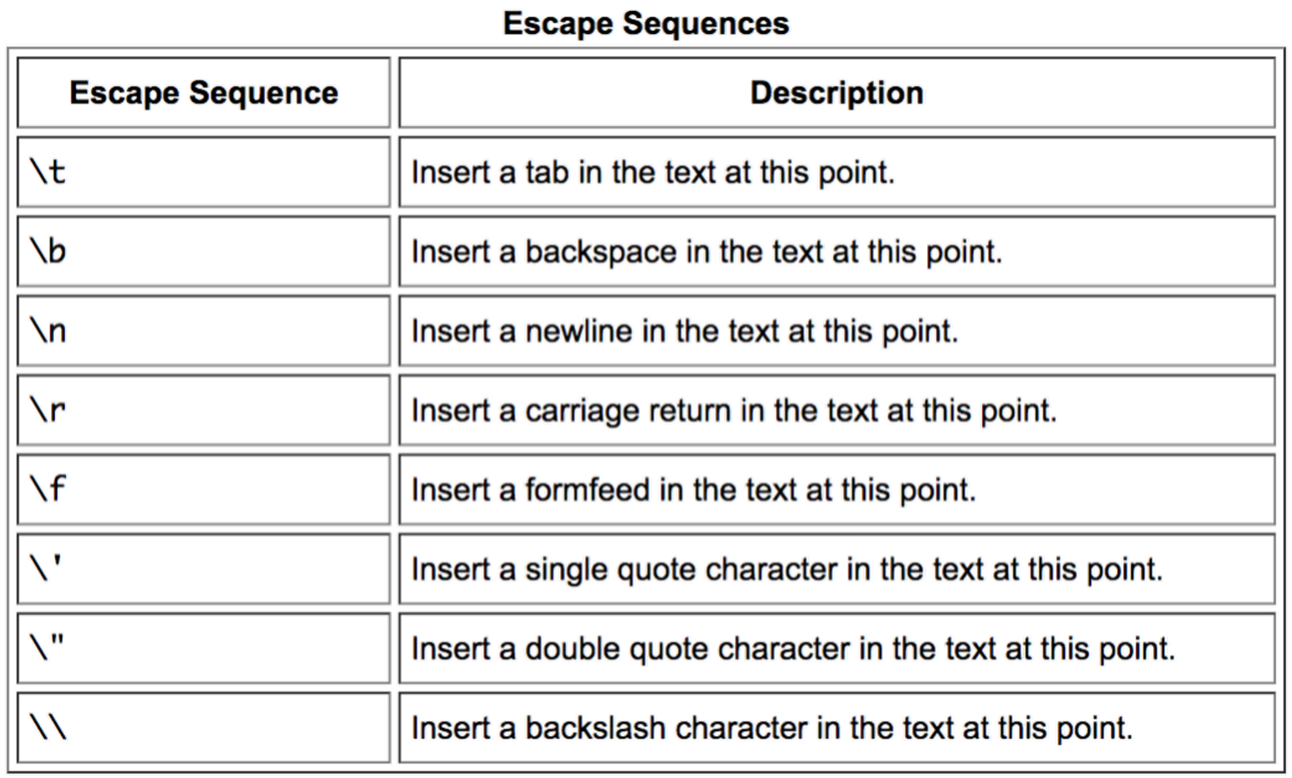
String



Expression	Value
“Hi, “ + “Bob”	“Hi, Bob”
“1” + “ 2 “ + “ 1”	“ 1 2 1”
“1234” + “ + “ + “99”	“1234 + 99”
“1234” + “99”	“123499”

Escape Sequences 转义字符序列

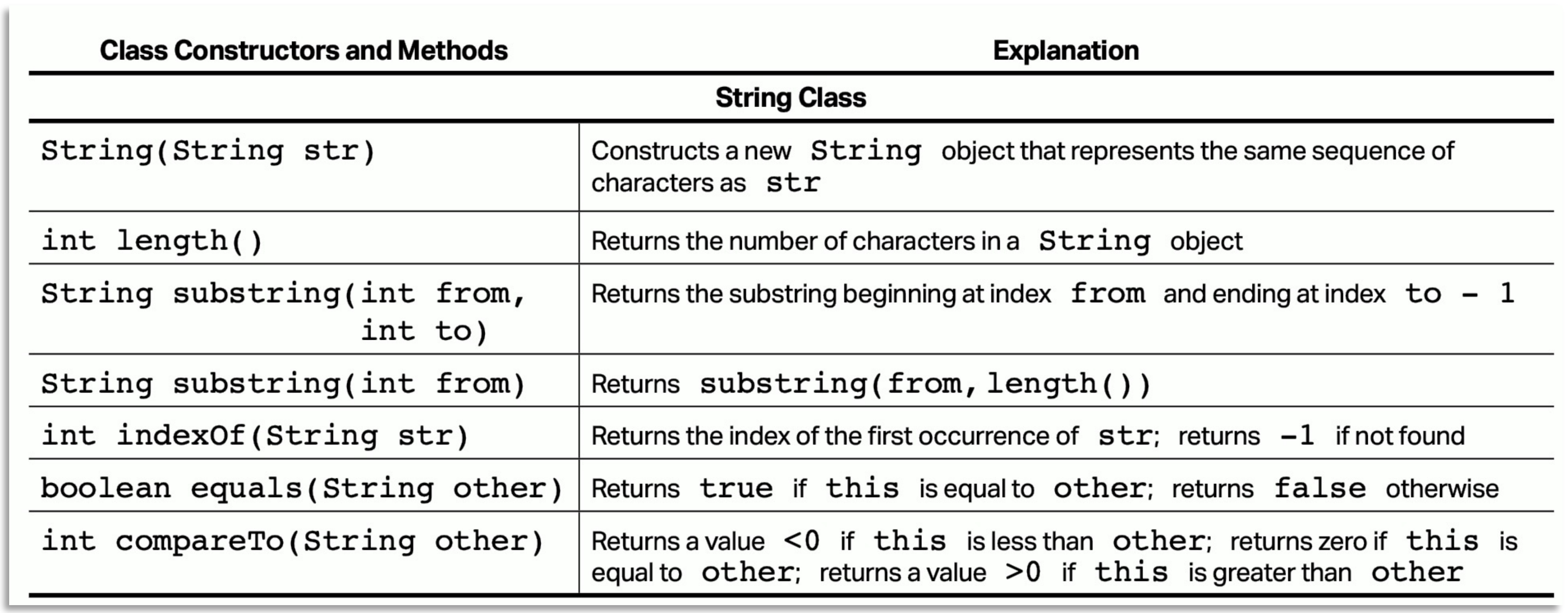
An escape sequence begins with the backslash character (\), and indicates that the character or characters that follow should be interpreted in a special way



Escape Sequences

Escape Sequence	Description
\t	Insert a tab in the text at this point.
\b	Insert a backspace in the text at this point.
\n	Insert a newline in the text at this point.
\r	Insert a carriage return in the text at this point.
\f	Insert a formfeed in the text at this point.
\'	Insert a single quote character in the text at this point.
\"	Insert a double quote character in the text at this point.
\\	Insert a backslash character in the text at this point.

String Methods



Class Constructors and Methods	Explanation
String Class	
`String(String str)`	Constructs a new `String` object that represents the same sequence of characters as `str`
`int length()`	Returns the number of characters in a `String` object
`String substring(int from, int to)`	Returns the substring beginning at index `from` and ending at index `to - 1`
`String substring(int from)`	Returns `substring(from, length())`
`int indexOf(String str)`	Returns the index of the first occurrence of `str`; returns `-1` if not found
`boolean equals(String other)`	Returns `true` if `this` is equal to `other`; returns `false` otherwise
`int compareTo(String other)`	Returns a value `<0` if `this` is less than `other`; returns zero if `this` is equal to `other`; returns a value `>0` if `this` is greater than `other`

1.2.1 Data types

1.2 Java Basics

1.2.2 Control Structure

1.2.2

Concept

Boolean Expressions 布尔表达式

Relational operators	Meaning	Boolean expressions	Result
<	Less than	`5 < 2`	`false`
>	Greater than	`5 > 2`	`true`
<=	Less than or equal to	`5 <= 2`	`false`
>=	Greater than or equal to	`5 >= 2`	`true`
==	Equal to	`5 == 2`	`false`
!=	Not equal to	`5 != 2`	`true`

The `if` Statement

- An **if-else** statement tells a program to do one thing if a condition is **true** and another thing if the condition is **false**.

```
if (Boolean condition)
{
   //statements
}
else
{
   //statements
}
```

Example

TESTDAILY

The if Statement

```
int score8 = 95;
if(score8 >= 90)
{
  System.out.println("watch TV for 2 hours");
}
else if(score8 >= 80)
{
  System.out.println("watch TV for 1.5 hours");
}
else if(score8 >= 70)
{
  System.out.println("watch TV for 1 hour");
}
else
{
  System.out.println("wash dishes");
}
```

Example

```
int score11 = 72;
if(score11 >= 60)
{
     if(score >= 90) // branch1: score >= 90
     {
          System.out.println("excellent");
     }
     else //branch2: 60 <= score <= 90
     {
          System.out.println("good");
     }
}
else // score <60
{
     if(score >= 40) //branch3: 40 <= score < 60
     {
          System.out.println("bad");
     }
     else  //branch4: score < 40
     {
          System.out.println("very bad");
     }
}
```

Logical Operation 逻辑操作符

&& (and 且)

|| (or 或)

! (not 非)

A	B	A && B
true	true	true
true	false	false
false	true	false
false	false	false

A	B	A || B
true	true	true
true	false	true
false	true	true
false	false	false

A	!A
true	false
false	true

Boolean Truth Tables 布尔真值表

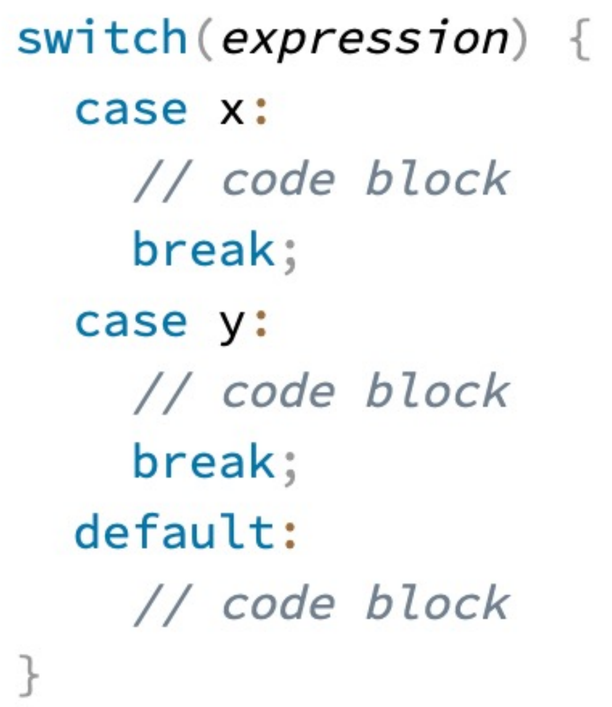
Example

Short-Circuited Evaluation can be used to avoid NullPointerException

```java
String str = null;
if(str != null && str.length() > 10)
{
  System.out.println("something");
}


// if(str.length() > 10 && str != null) // Error: you must check null first
```

switch

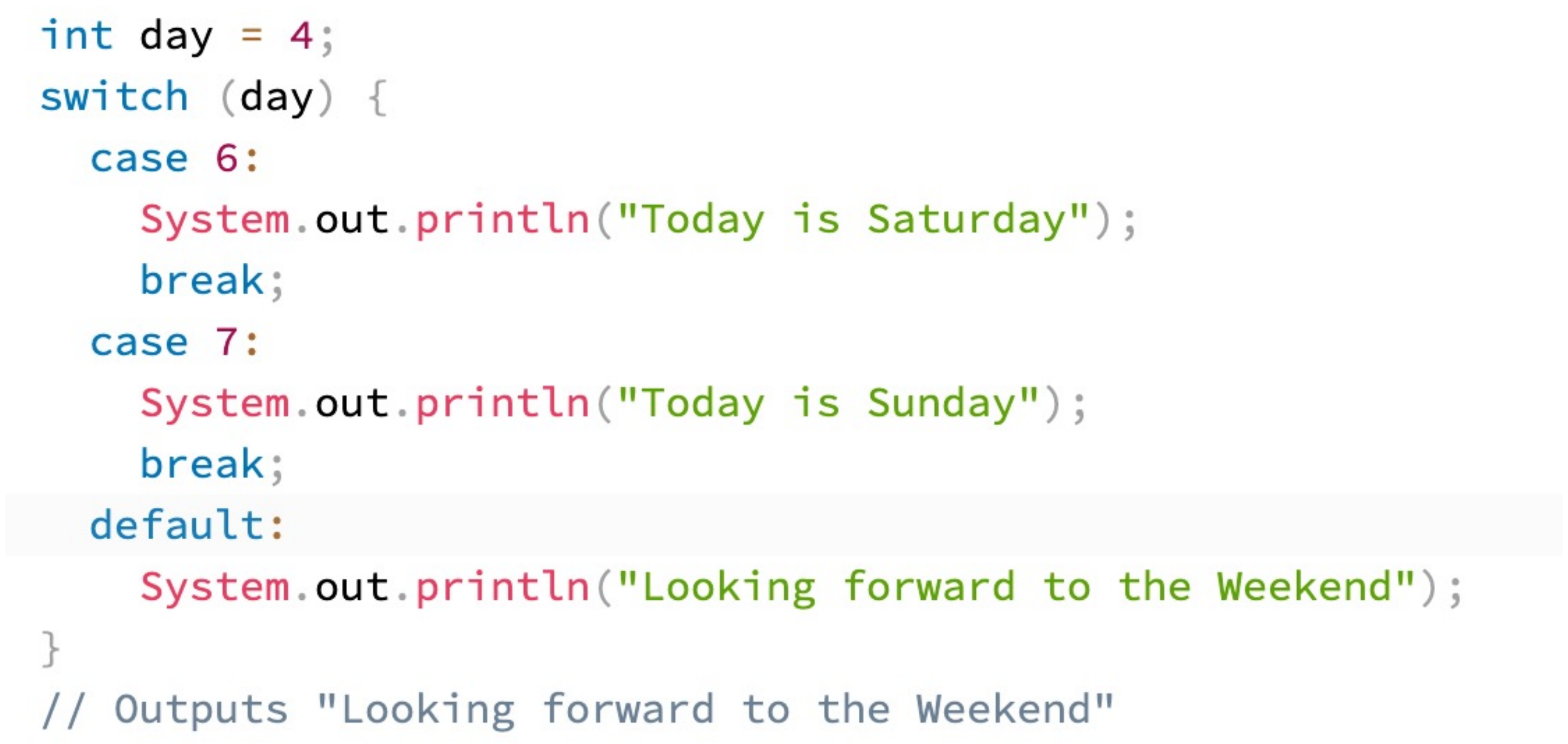
switch operator allows program to execute one or more case block



```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

https://www.w3schools.com/java/java_switch.asp

switch



```java
int day = 4;
switch (day) {
  case 6:
    System.out.println("Today is Saturday");
    break;
  case 7:
    System.out.println("Today is Sunday");
    break;
  default:
    System.out.println("Looking forward to the Weekend");
}
// Outputs "Looking forward to the Weekend"
```

https://www.w3schools.com/java/java_switch.asp

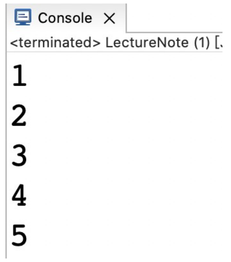
Loops

- A loop executes instructions repeatedly while a condition is true
  - while loops
  - for loops

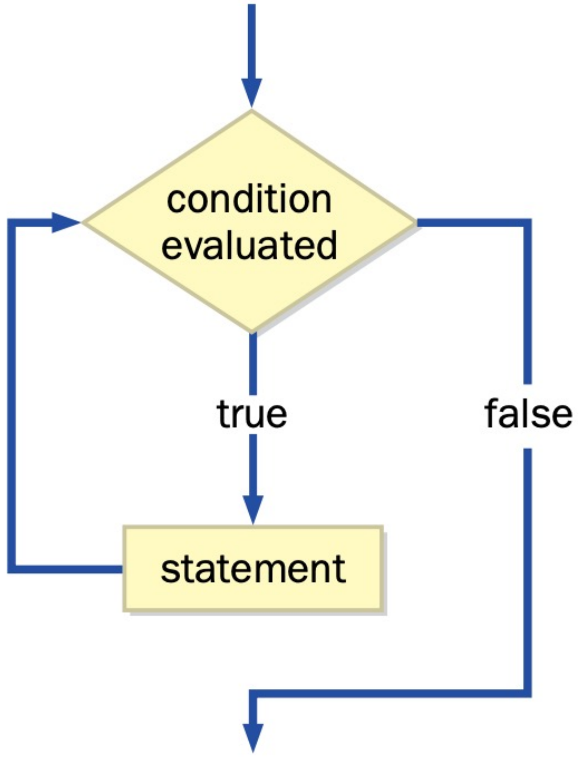
Example

While loops

```java
int count = 1;
while (count <= 5)
{
     System.out.println(count);
     count = count + 1;
}
```


Console ✕
<terminated> LectureNote (1) [.

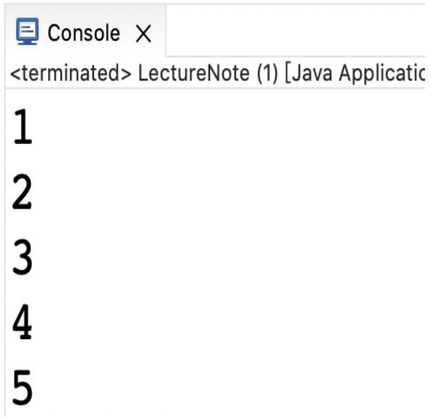
```
1
2
3
4
5
```

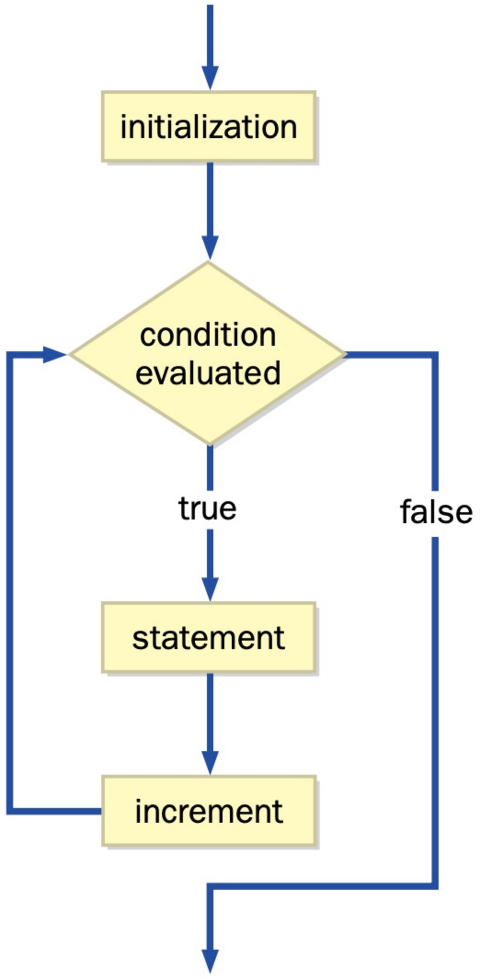


Example

for Loops

```
for (int count=1; count <= 5; count++)
            System.out.println (count);
```


```
Console X
<terminated> LectureNote (1) [Java Applicatic
1
2
3
4
5
```



Nested Loops

- The body of a loop contains another loop
- Each time the outer loop executes once, the inner loop executes completely

```
while (…)
{
    while (…)
    {

    }
}
```

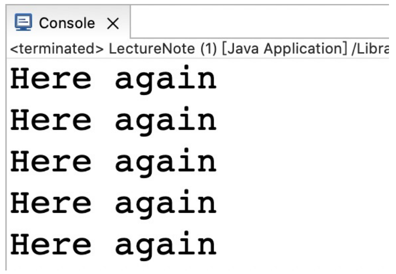
```
for (…)
{
    for (…)
    {

    }
}
```

Example

Nested Loops

How many times does the string "Here again" get printed?

```
int count1, count2;
count1 = 1;
while (count1 <= 10)
{
    count2 = 1;
    while (count2 <= 50)
    {
        System.out.println("Here again");
        count2++;
    }
    count1++;
}
```



```
Here again
Here again
Here again
Here again
Here again
```

1.2.2 Control Structure

1. if Statement
2. while, for loops

Unit1 Introduction

1. Introduction to Data Structure
2. Java Basics