Java Basic Program

Module 1

1. Introduction part, Languages Programing, Java history, JDK, “hello World”
2. Project, package, Class, method
3. Variables. Keyboard input
4. Variables and data types
5. Consultation

Module 2

1. TEST #1 and second part Compilation and constructor
2. Methods and Random ways
3. Practice and examples with methods
4. Boolean. Boolean expressions
5. Consultation

Module 3

1. if-else-if
2. Switch, ternary operator
3. Loops, for
4. Loops, while, do while
5. Consultation

Module 4

1. Arrays in Java
2. Arrays search and sort
3. String, StringBuilder, StringBuffer, practice
4. TEST #2 and second part Class and Object
5. Consultation

Module 5 (Optional)

1. Method main() for the test and introduction to JUnit testing
2. Practice, repetitions, console Lottery game
3. Practice, implementation of the distribution of cards in Poker
4. Summarizing and Introduction to the professional course program and the profession Back-end developer in Java
5. Consultation (questions)

Java Basic Program

Module 1

1. Introduction part, Languages Programing, Java history, JDK, “hello World”

2. Project, package, Class, method

3. Variables. Keyboard input

4. Variables and data types

INTRODUCTION TO

JAVA

MATERIALS AND RESOURCES

- Books
  - Head First Java (2nd edition) - Bert Bates, Kathy Sierra
  - Grokking Algorithms - Aditya Y. Bhargava
  - Thinking in Java (4th edition) - Bruce Eckel
- Documentation & help
  - Oracle's Java Tutorial: https://docs.oracle.com/javase/tutorial/
  - Q&A: https://stackoverflow.com/questions/tagged/java

GETTING STARTED

- Toolbox
  - **JDK 19**
    https://www.oracle.com/java/technologies/javase/jdk19-archive-downloads.html
  - **Intellij IDEA Community Edition:**
    https://www.jetbrains.com/idea/download/
  - **Git:** https://git-scm.com/downloads
- Collaboration
  - **GitHub:** https://github.com/
- Communication
  - **Telegram**

GETTING STARTED

Java developer

1. Java от 8 до 17
2. Database
3. Spring as framework
4. Git
5. Unix systems
6. …

QA engineer

1. Java Basic
2. Database
3. Tools for test
4. Git
5. Unix systems
6. …

Front-end developer

1. Java Basic
2. Database
3. JavaScript, React or Angular
4. Git
5. Unix systems
6. …

Классификация

1. (стажеры, Trainee), которые знают основы Java, но пока не умеют применять их на практике.
2. (младший разработчик, Junior). Имеет минимальный опыт работы, его работу проверяют коллеги.
3. (Java Middle). Этот специалист уже имеет опыт работы, он может самостоятельно писать код, который не нуждается в проверке, работает с заданиями средней сложности.
4. (старший разработчик, Senior). Легко справляется с любыми задачами, несет полную ответственность за сроки, написанный код, внесенные изменения.
5. (TeamLead, TeachLead). Специалист высокого уровня, который не только занимается написанием кода, но и раздает ТЗ, контролирует работу младших сотрудников.
6. (Architect). Самый высокооплачиваемый специалист, занимающийся проектированием, который может самостоятельно вытянуть весь проект

ПЛЮСЫ И МИНУСЫ

Плюсы

1. Вакансий Java-программистов много, поэтому найти работу не проблема.
2. Профессию можно освоить примерно за 1 год на курсах.
3. Возможность работать удаленно.
4. Перспективное профессиональное направление.
5. Работа в творческой команде.
6. Стабильная, высокая зарплата.

Минусы

1. Большая ответственность.
2. Без опыта сложно устроиться на работу.
3. Болят пальцы.

Личные качества

1. Целеустремленность.
2. Сосредоточенность.
3. Аккуратность.
4. Пунктуальность.
5. Желание самообучаться.

JAVA OVERVIEW

BRIEF HISTORY

- **Green Team** (James Gosling, Mike Sheridan, Patrick Naughton)
- Initiated language project in June 1991
- Originally designed for small embedded systems in electronic appliances
- Initially called **Greentalk**, shortly after renamed to **Oak**
- A few years later renamed to Java

REASONS BEHIND NAMING

- **Reflection** of the **essence** of the technology
  - Revolutionary and dynamic
  - Easy to spell and fun to say
- Suggested names were **Dynamic**, **Silk**, **Jolt**, **DNA**
- **Java** is an island where first coffee was produced

PRINCIPLES

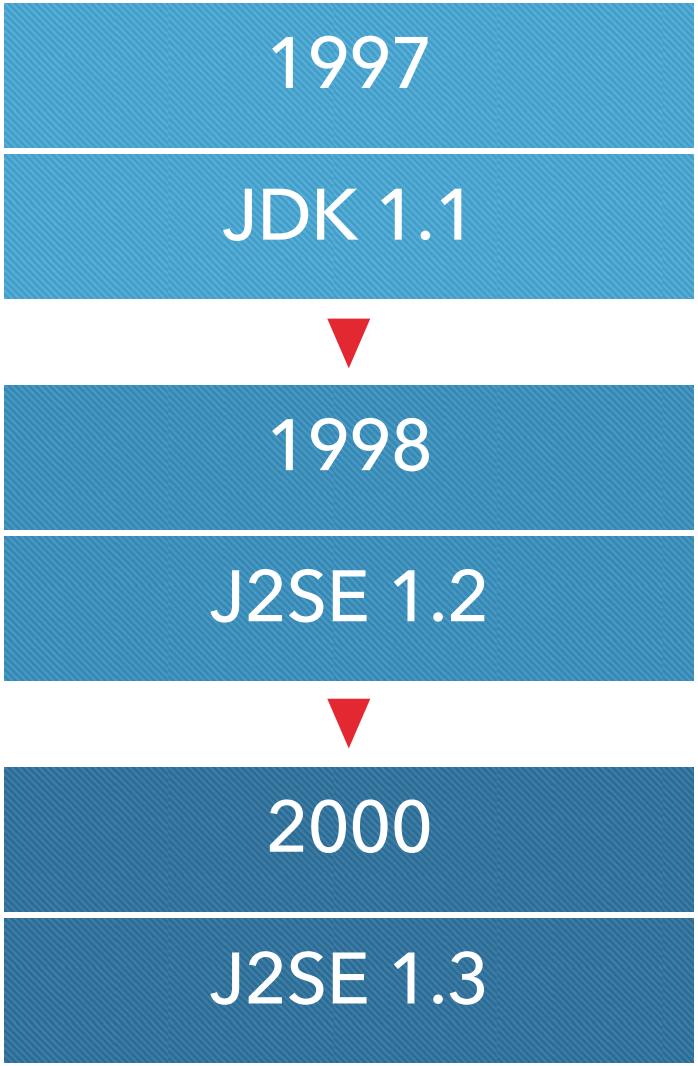
- Simple and robust
- Portable and platform independent
- High performant, multithreaded and secured
- Following object oriented language features

WHAT JAVA IS USED FOR

- Mobile phones
  - Google's **Android** OS
- Point of sale systems
  - **McDonald's** checkout computers
- Video games
  - **Minecraft** - one of the most popular games of all time
- Trading applications
  - **Murex** - application used for front to back connectivity is utilized by many banks
- Big data technologies
  - **Elasticsearch** - distibuted search and analytics engine

RELEASE TIME LAPSE

- JDK 1.1 Major Changes
  - **Extensive retooling** for AWT event model
  - Introduced **inner classes**
  - Addition of **JDBC** and **JavaBeans**
- J2SE 1.2 Major Changes
  - **Rebranded** as Java 2
- J2SE 1.3 Major Changes
  - Bundled with **Hotspot JVM**
  - **JavaSound**, Java Naming and Directory Interface
  - Java Platform **Debugger** Architecture



RELEASE TIME LAPSE

- JDK 1.4 Major Changes
  - First release developed under **Java Community Process**
  - Inclusion of **regular expressions**
- J2SE 5.0 Major Changes
  - For-each loop
  - Generics & Auto-boxing
  - Var-args
- Java SE 6 Major Changes
  - **Bundled** with database manager
  - **Facilitates** use of scripting languages with JVM

RELEASE TIME LAPSE

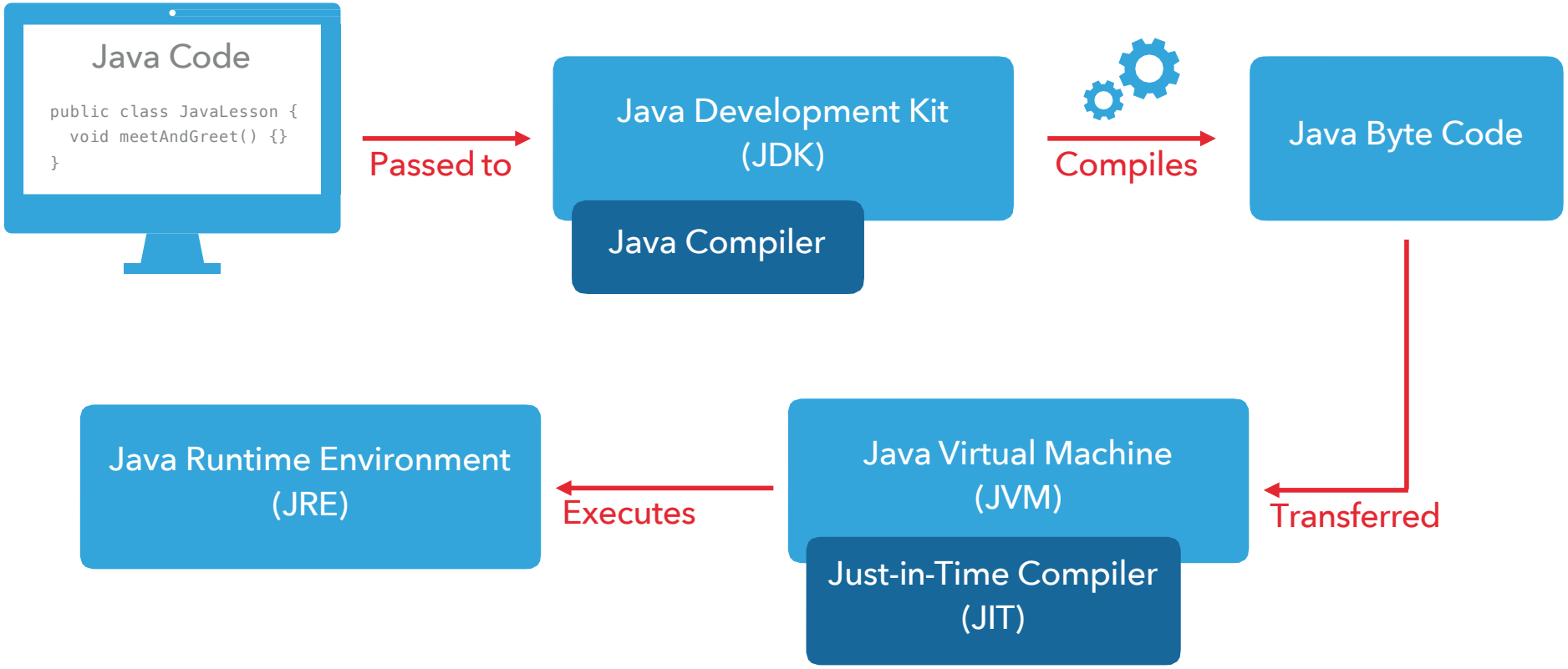
- Java SE 7 Major Changes
  - Strings in **switch**
  - JVM **support** for dynamic languages
- Java SE 8 Major Changes
  - **Lambda expressions** & Streams API
  - Default methods
  - **Reworked** dates API
- JDK 9 Major Changes
  - Module system
  - **Extended** Streams API

Java and evolution

- James Gosling, Mike Sheridan, Patrick Naughton
- Green Team
- Greentalk .gt
- Oak
- Java
- Sun Microsystem
- JDK 1.0 был выпущен 23 января 1996 года.
- Oracle Corporation

JDK Альфа и Бета (1995)
JDK 1.0 (23 января 1996 г.)
JDK 1.1 (19 февраля 1997 г.)
J2SE 1.2 (8 декабря 1998 г.)
J2SE 1.3 (8 мая 2000 г.)
J2SE 1.4 (6 февраля 2002 г.)
J2SE 5.0 (30 сентября 2004 г.)
Java SE 6 (11 декабря 2006 г.)
Java SE 7 (28 июля 2011 г.)
Java SE 8 (18 марта 2014 г.)
Java SE 9 (21 сентября 2017 г.)
Java SE 10 (20 марта 2018 г.)
Java SE 11 (сентябрь 2018 г.)
Java SE 12 (март 2019 г.)
Java SE 13 (сентябрь 2019 г.)
Java SE 14 (март 2020 г.)
Java SE 15 (сентябрь 2020 г.)
Java SE 16 (март 2021 г.)
Java SE 17 (сентябрь 2021 г.)
Java SE 18 (будет выпущен к марту 2022 г.)

INTERNAL WORKFLOW



JAVA COMPONENTS: JVM & JRE & JDK

- Java Virtual Machine (**JVM**) is an abstract computing machine that enables computer to run a Java program
- Java Runtime Environment (**JRE**) is a software package that contains what is required to run a Java program. It includes a Java Virtual Machine implementation together with an implementation of the Java Class Library
- Java Development Kit (**JDK**) is a superset of a JRE and contains tools for Java programmers, e.g. a java compiler

JAVA COMPONENTS: JVM & JRE & JDK

- Виртуальная машина Java (JVM) – это абстрактная вычислительная машина, которая позволяет компьютеру запускать программу Java.
- Java Runtime Environment (JRE) – это программный пакет, содержащий все необходимое для запуска программы Java. Он включает в себя реализацию виртуальной машины Java вместе с реализацией библиотеки классов Java.
- Комплект для разработки Java (JDK) представляет собой расширенный набор JRE и содержит инструменты для программистов на Java, например. компилятор Java

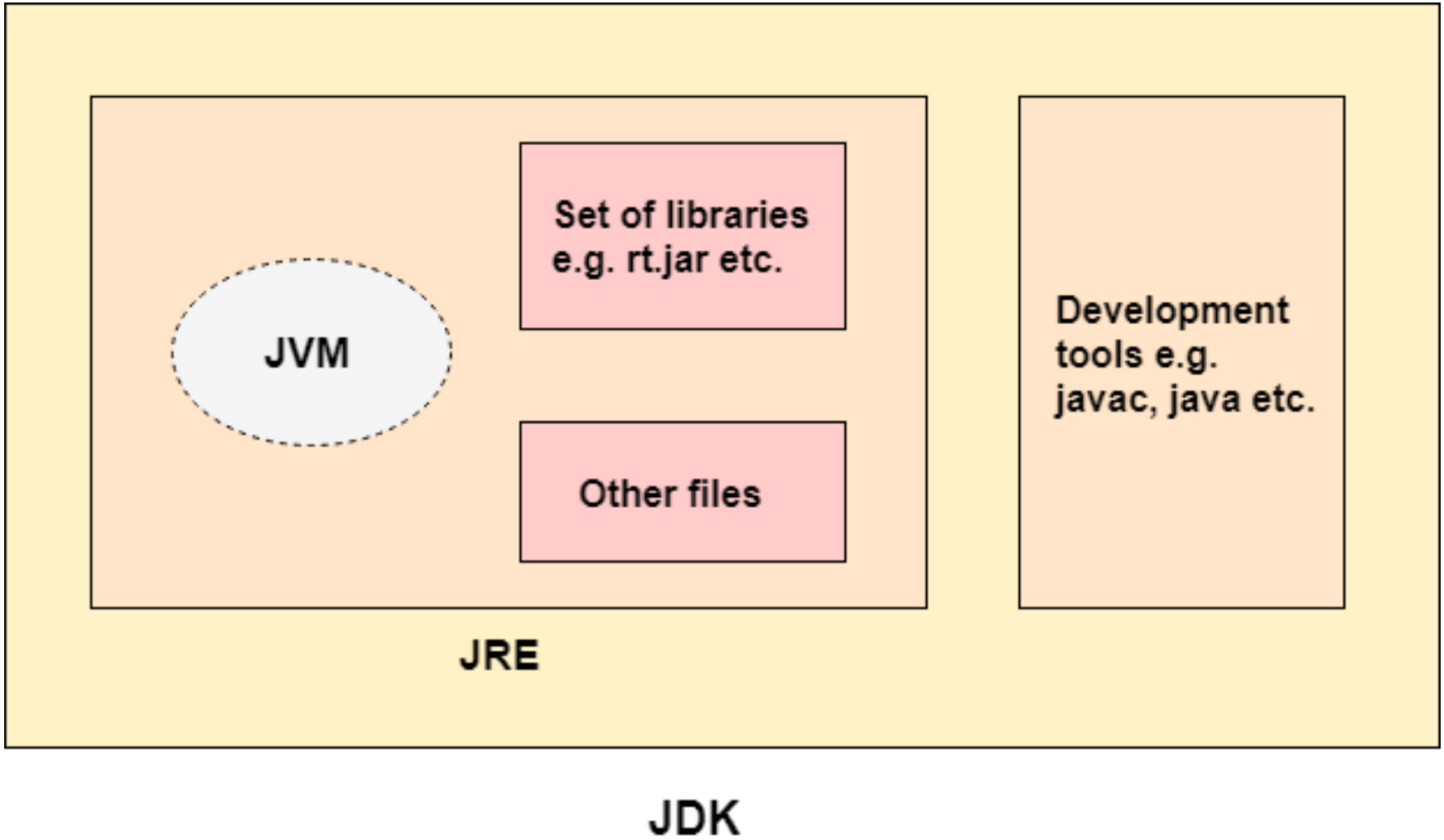
JAVA COMPONENTS: JVM & JRE & JDK

JVM (Java Virtual Machine) - это виртуальная машина Java - основная часть исполняющей системы Java, так называемой Java Runtime Environment (JRE). Виртуальная машина Java исполняет байт-код Java, предварительно созданный из исходного текста Java-программы компилятором Java (javac). JVM обеспечивает платформо-независимый способ выполнения кода. Программисты могут писать код не задумываясь, как и где он будет выполняться.

JRE (Java Runtime Environment) - это минимальная реализация виртуальной машины, необходимая для исполнения Java приложений, без компилятора и других средств разработки. Состоит из виртуальной машины и библиотек Java классов.

JDK (Java Development Kit) - это комплект разработчика приложений на языке Java, включающий в себя компилятор, стандартные библиотеки классов Java, примеры, документацию, различные утилиты и исполнительную систему JRE.

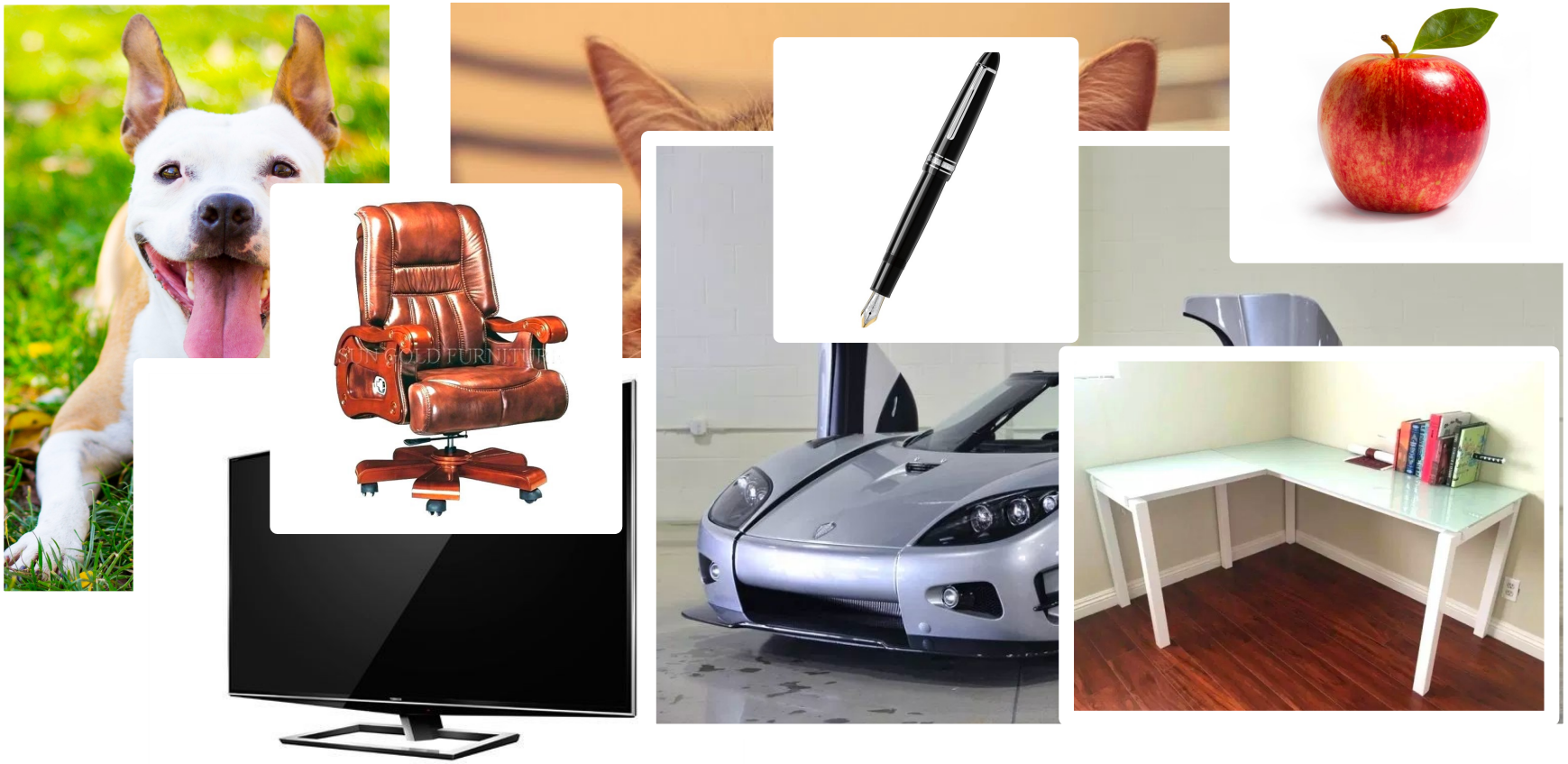
JAVA COMPONENTS: JVM & JRE & JDK



OBJECT-ORIENTED PRINCIPLES OVERVIEW

WHAT IS AN OBJECT?

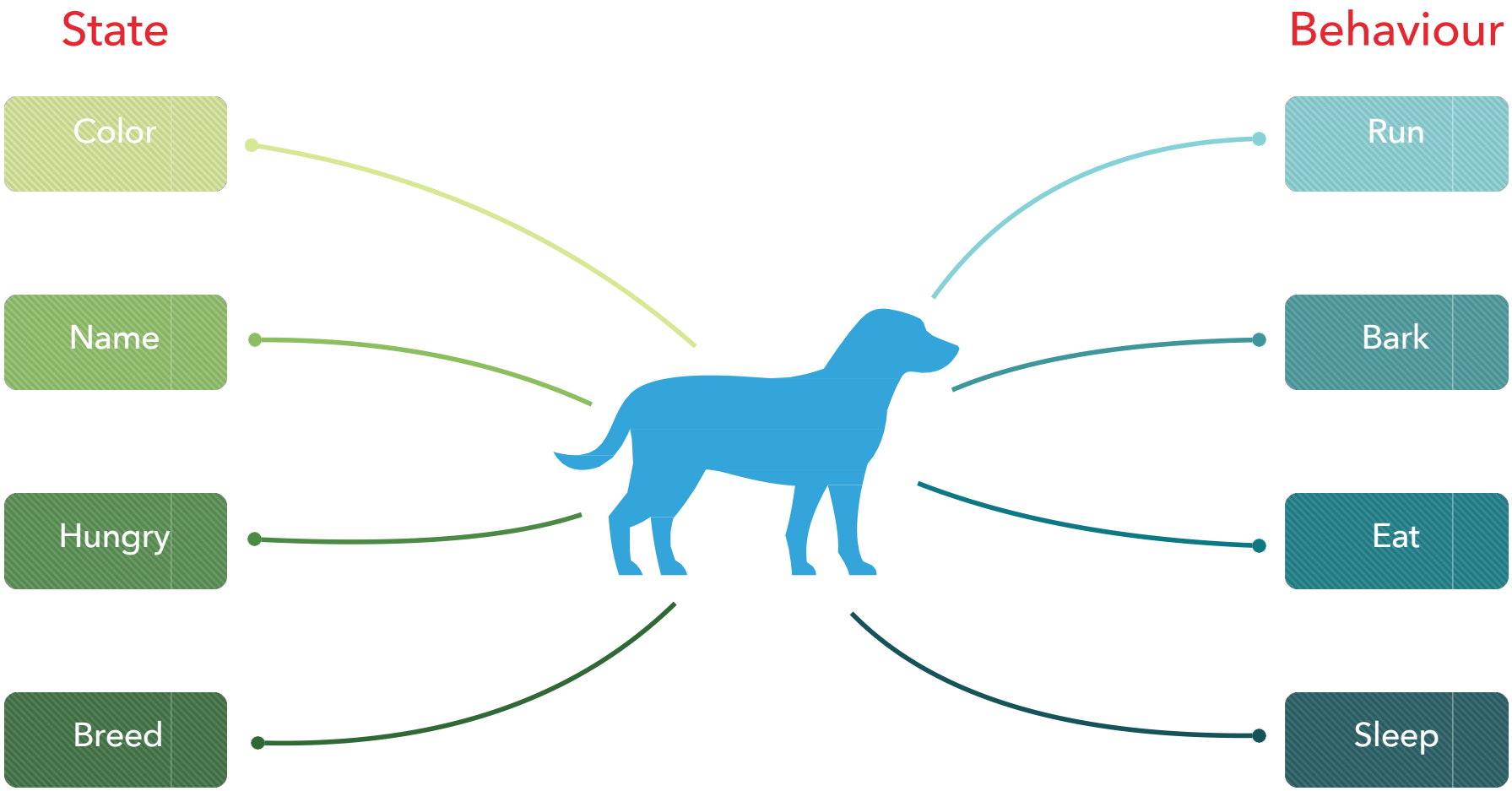
- Just look around... Everything is an object!



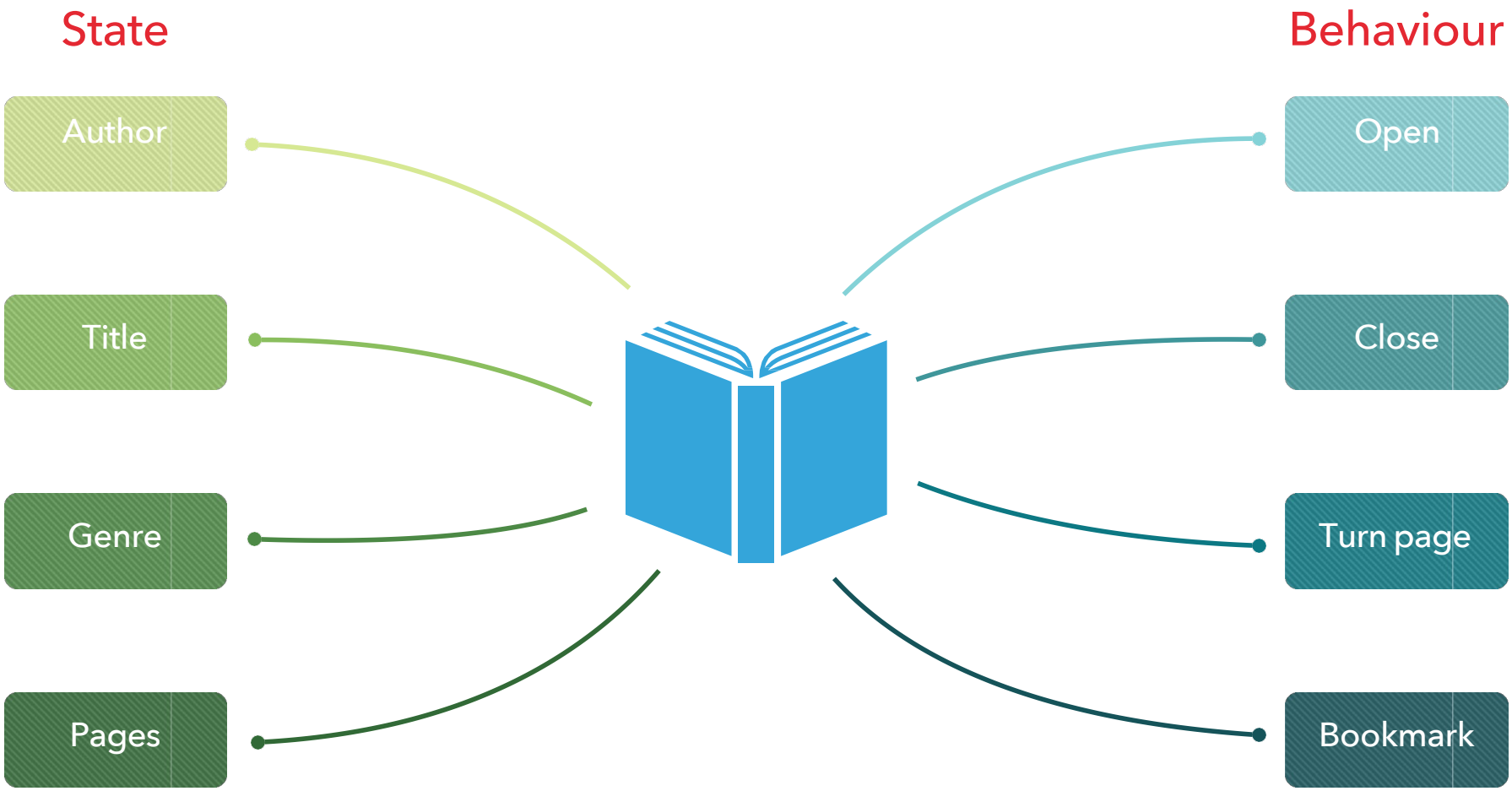
CHARACTERISTICS

- Every object share two characteristics:
  - Object has some sort of state
  - Object might have behaviour

OBJECT BREAKDOWN: DOG



OBJECT BREAKDOWN: BOOK



ALGORITHMS

DEFINITION OF AN ALGORITHM

- An algorithm is a procedure or formula for solving a problem, based on conducting a sequence of specified actions
  - In mathematics and computer science, an algorithm usually means a small procedure that solves a recurrent problem
- A computer program can be viewed as an elaborate algorithm

ALGORITHM BREAKDOWN: TAXI RIDE

1. Contact with the taxi service
2. Provide pickup & destination addresses
3. Wait for the cab
4. Get in the car
5. Enjoy your ride
6. Pay for the service upon arrival

ALGORITHM BREAKDOWN: PLANT A PLANT

1. Dig a hole
2. Put roots into soil
3. Cover the hole
4. If soil is wet then end
5. Water the plant

TASK 1: OBJECTIVES

1. The program should define **two** integer variables
2. **Calculate the sum** of these variables
3. **Print** result to the console

TASK 1: CODE LISTING

1. Create 1st variable and assign it value of 10
2. Create 2nd variable and assign it value of 20
3. Assign 3rd variable result of computed sum
4. Print 3rd variable to the console

```
package lv.javalessons.demo;

public class CalculatorSumTest {

    public static void main(String[] args) {

      1 int firstNumber = 10;
      2 int secondNumber = 20;
      3 int sumResult = firstNumber + secondNumber;
      4 System.out.println(sumResult);

    }

}
```

TASK 2: OBJECTIVES

1. The program should create random number generator
2. Generate random number within 0 - 100 range inclusive
3. Print result to the console

TASK 2: CODE LISTING

1. Create random generator and assign it to the variable
2. Generate random number and assign result to the variable
3. Print *randomNumber* variable to the console

```
package lv.javalessons.demo;

import java.util.Random;

public class RandomNumberGeneratorTest {

    public static void main(String[] args) {
      1 Random randomGenerator = new Random();
2       int randomNumber =
randomGenerator.nextInt(101);
      3 System.out.println(randomNumber);
    }

}
```

WRITING OUTPUT TO THE CONSOLE

WRITING OUTPUT TO THE CONSOLE: SYNTAX

- Write to the console content of the *"args"* variable

```
System.out.println(args);
```

- Write to the console directly without variable

```
System.out.println("Hello World");
```

WRITING OUTPUT TO THE CONSOLE: EXAMPLE

Printing Variable

```
int a = 10;
String b = "Hi"
System.out.println(a);
System.out.println(b);
```

```
10
Hi

Process finished with exit
code 0
```

Printing directly

```
System.out.println(374);
System.out.println("A");
```

```
374
A

Process finished with exit
code 0
```

WRITING OUTPUT STATEMENT BREAKDOWN

Class with standard output, input and error output streams

Function that writes argument to the specified stream

```
System.out.println("Hello World");
```

Argument to be written to the stream

Inner class that helps to specify stream flow direction

INTRODUCTION TO JAVA

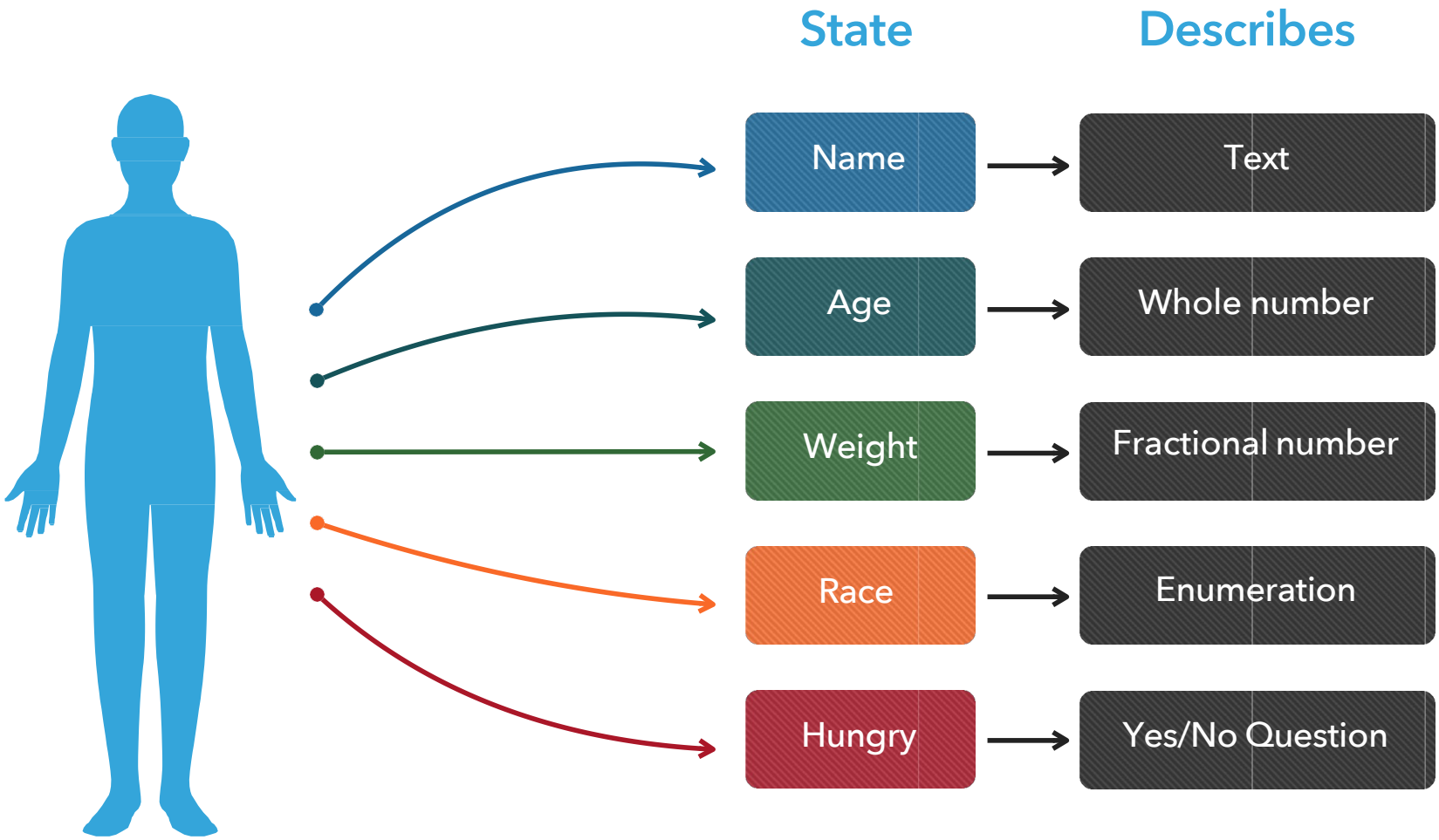
LESSON

VARIABLES OVERVIEW

VARIABLES DEFINITION

- Variable is a named placeholder that
  - Stores data
  - Describes what type of data you can store
  - Describes size or amount of data it can store

VARIABLES RELATIONSHIP WITH OBJECTS



DATA TYPE CATEGORIES

- Primitive values
  - Integer: *byte, short, int, long* (e.g. 3, 7, 42, 2018)
  - Fractional: *float, double* (e.g. 3.1415, 2.7, 19.0)
  - Logical: *boolean* (true or false)
  - Textual: *char* (e.g. a, b, c, x, y, z)
- Reference values
  - Everything else

PRIMITIVE DATA TYPES IN DEPTH: INTEGER

Name	Assignable Values	Space
byte	-128 … 127	1 byte
short	-32,768 … 32,767	2 bytes
int	$-2^{31} \ldots 2^{31}-1$	4 bytes
long	$-2^{63} \ldots 2^{63}-1$	8 bytes

PRIMITIVE DATA TYPES IN DEPTH: FLOATING POINT

Name	Precision	Space
float	Single	4 bytes
double	Double	8 bytes

PRIMITIVE DATA TYPES IN DEPTH: LOGICAL

Name	Assignable Values	Space
boolean	true / false	1 byte

PRIMITIVE DATA TYPES IN DEPTH: TEXTUAL

Name	Assignable Values	Space
char (unicode)	0 ('\u0000') ... 65535 ('\uffff')	2 bytes

DEFINING VARIABLES IN JAVA CODE

VARIABLE DECLARATION IN JAVA: SYNTAX

- Variable declaration without value assignment

```
type name;
```

- Variable declaration **with** value assignment

```
type name = value;
```

VARIABLE DECLARATION IN JAVA: EXAMPLE

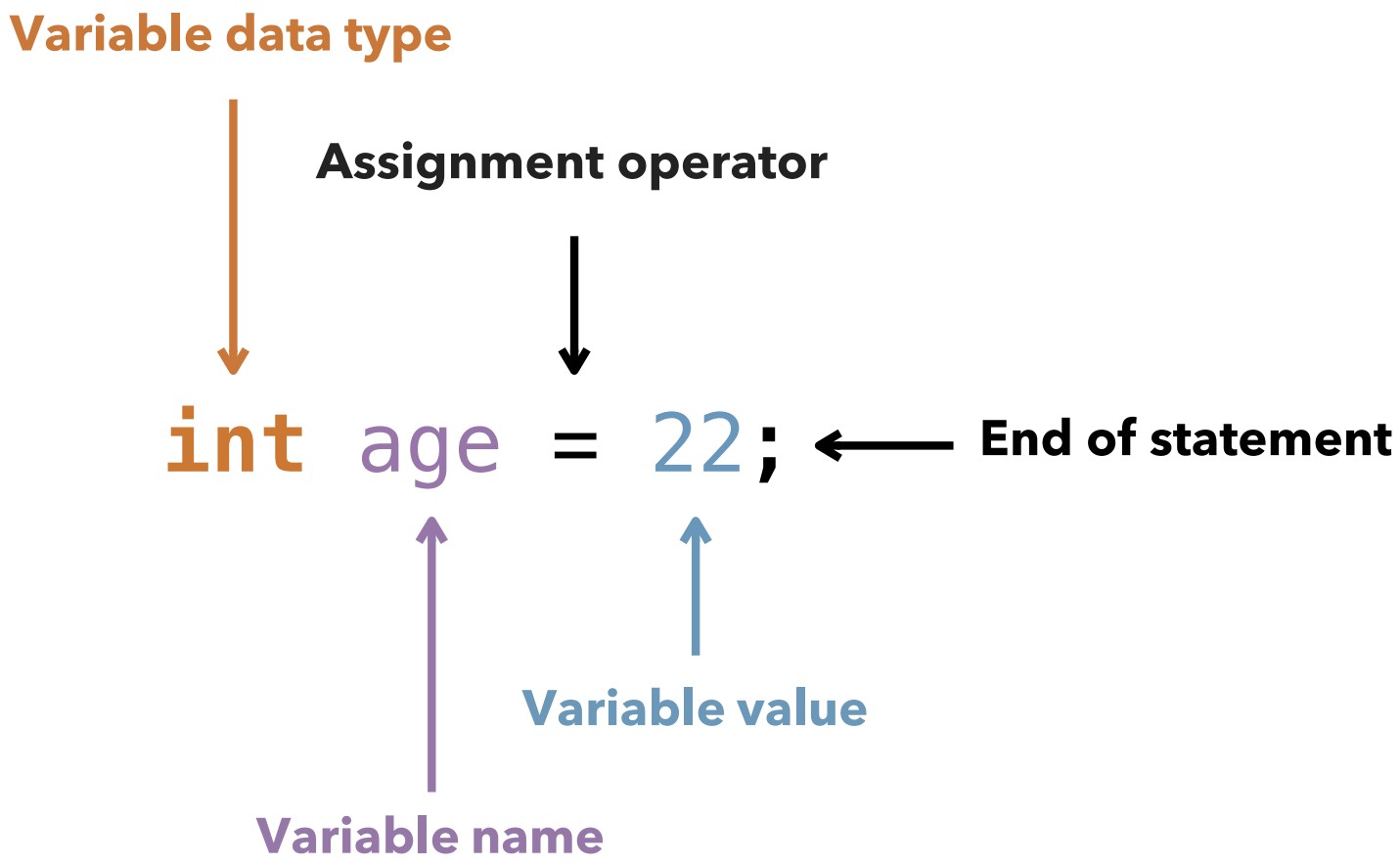
- Variable declaration without value assignment

```
int age;
```

- Variable declaration **with** value assignment

```
int age = 22;
```

VARIABLE DECLARATION BREAKDOWN



MORE EXAMPLES

```
byte numberOfWheels = 4;
short selfEsteem = -1;
int studentsGraduated = 1001;
long height = 80;
float pie = 3.14f;
double weight = 70.5;
boolean hungry = true;
char lastLetterOfTheAlphabet = 'Z';
```

NAMING RULES

- Any variable is allowed to **start** with
  - Letters (A-Z)
  - Special characters ('$' - dollar, '_' - underscore)
- Any variable name is allowed to **contain**
  - Alphanumeric characters (A-Z, 0-9)
  - Special characters ('$' - dollar, '_' - underscore)
- Variable name is **case-sensitive**
- Java language keywords[1] or reserved words **cannot** be used as variable name

[1] List of keywords can be found at https://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html

NAMING DOS

- Single-worded name should be **lowercase**
- Multi-worded name should
  - First word **lowercase**
  - **Subsequent words start with** capital letters
  - No intervening spaces or punctuation
- Explains the purpose of variable

NAMING DON'TS

- Starting variable name with $ or _ is **highly discouraged**
- **Avoid** using $ anywhere in the variable name

NAMING DOS AND DON'TS EXAMPLES

- Please, do
  - size, xCoordinate, skinColor, currentDayOfTheWeek
- Please, don't
  - _counter, $bankBalance, Timestamp, 7daysOfTheWeek, !variableName, *notPointer

ARITHMETIC OPERATORS

ARITHMETIC OPERATORS OVERVIEW

Operator	Operation
+	Addition
-	Subtraction
/	Division
*	Multiplication
%	Remainder

OPERATORS BREAKDOWN: ADDITION

Integer numbers

```
int a = 10;
int b = 30;
int result = a + b;
```

```
result == 40
```

Fractional numbers

```
double x = 1.5;
double y = 2.7;
double result = x + y;
```

```
result == 4.2
```

OPERATORS BREAKDOWN: SUBTRACTION

Integer numbers

```
int a = 30;
int b = 20;
int result = a - b;
```

```
result == 10
```

Fractional numbers

```
double x = 5.4;
double y = 1.6;
double result = x - y;
```

```
result == 3.8
```

OPERATORS BREAKDOWN: MULTIPLICATION

Integer numbers

```
int a = 2;
int b = 4;
int result = a * b;
```

```
result == 8
```

Fractional numbers

```
double x = 2.5;
double y = 6.4;
double result = x * y;
```

```
result == 16.0
```

OPERATORS BREAKDOWN: DIVISION

Integer numbers

```
int a = 10;
int b = 5;
int result = a / b;
```

```
result == 2
```

Fractional numbers

```
double x = 18.0;
double y = 4.8;
double result = x / y;
```

```
result == 3.75
```

OPERATORS BREAKDOWN: REMAINDER

Integer numbers

```
int a = 9;
int b = 6;
int result = a % b;
```

```
result == 3
```

Fractional numbers

```
double x = 10.0;
double y = 4.5;
double result = x % y;
```

```
result == 1.0
```

TRICKY QUESTIONS

Type for division result is integer?

```
int a = 10;
int b = 4;
int result = a / b;

result == ?
```

Type for division result is double?

```
int x = 10;
int y = 4;
double result = x / y;

result == ?
```

TYPE CONVERSION: CASTING

- Operations with widening result require explicit type conversion (cast)

```
int    x  =  10;
int    y  =  4;
double result = x / (double) y;

result == ?
```

COMMON MISTAKES AND PITFALLS OVERVIEW

OVERVIEW

1. Missing terminator sign ';'
2. Incorrect spelling
    1. Class name
    2. Package name
    3. Variable name
3. Code placement outside of the body
4. Missing quotes or misplacement

(1) MISSING TERMINATOR SIGN: THE CODE

```
public class ForgotSemicolonAgain {

    public static void main(String[] args) {
        System.out.println("Oops.. I did it again")
    }

}
```

(1) MISSING TERMINATOR SIGN: THE FIX

```
public class ForgotSemicolonAgain {

    public static void main(String[] args) {
        System.out.println("Oops.. I did it again");
    }

}
```

(2.1) BAD CLASS SPELLING: THE CODE

```
public class sizeMatters {

    public static void main(String[] args) {
        system.out.println("Sorry, it does");
    }

}
```

(2.1) BAD CLASS SPELLING: THE FIX

```
public class SizeMatters {

    public static void main(String[] args) {
        System.out.println("Sorry, it does");
    }

}
```

(2.2) BAD PACKAGE SPELLING: THE CODE

```
package lv.javajava.lessons.HOMEWork;
```

(2.2) BAD PACKAGE SPELLING: THE CODE

```
package lv.javajava.lessons.homework;
```

(3) CODE PLACEMENT OUTSIDE OF THE BODY: THE CODE

```
public class AttentionPlease {

    System.out.println("Hide and seek");
    public static void main(String[] args) {
    }

}
```

(3) CODE PLACEMENT OUTSIDE OF THE BODY: THE FIX

```
public class AttentionPlease {

    public static void main(String[] args) {
        System.out.println("Hide and seek");
    }

}
```

(4) QUOTES MISPLACEMENT: THE CODE

```
public class NoSleepNoFocus {

    public static void main(String[] args) {
        System.out.println(I wanna coffee);
        System.out.println("So bad);
    }

}
```

(4) QUOTES MISPLACEMENT: THE FIX

```
public class NoSleepNoFocus {

    public static void main(String[]args)  {
        System.out.println("I wanna coffee");
        System.out.println("Sobad");
    }

}
```