CS121 Data Structures
Introduction

Monika Stepanyan
mstepanyan@aua.am

Spring 2024

Important Data and Statistics

- 30 classes remaining
- 53 days till the Midterm exam I
- 44 days till the Spring Break

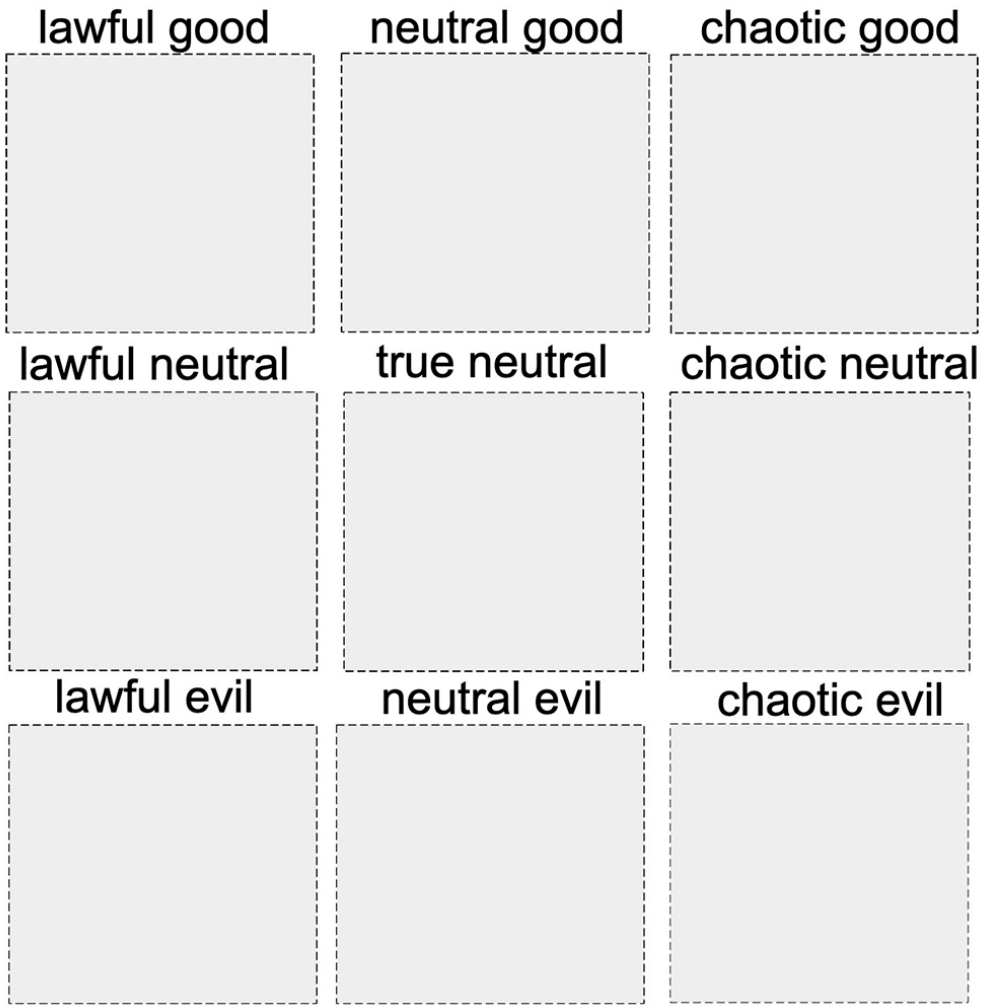
Important Data and Statistics

Please don't call me:

- Ynker; as I am not a tovarisch (comrade)
- Professor; as I am not a professor

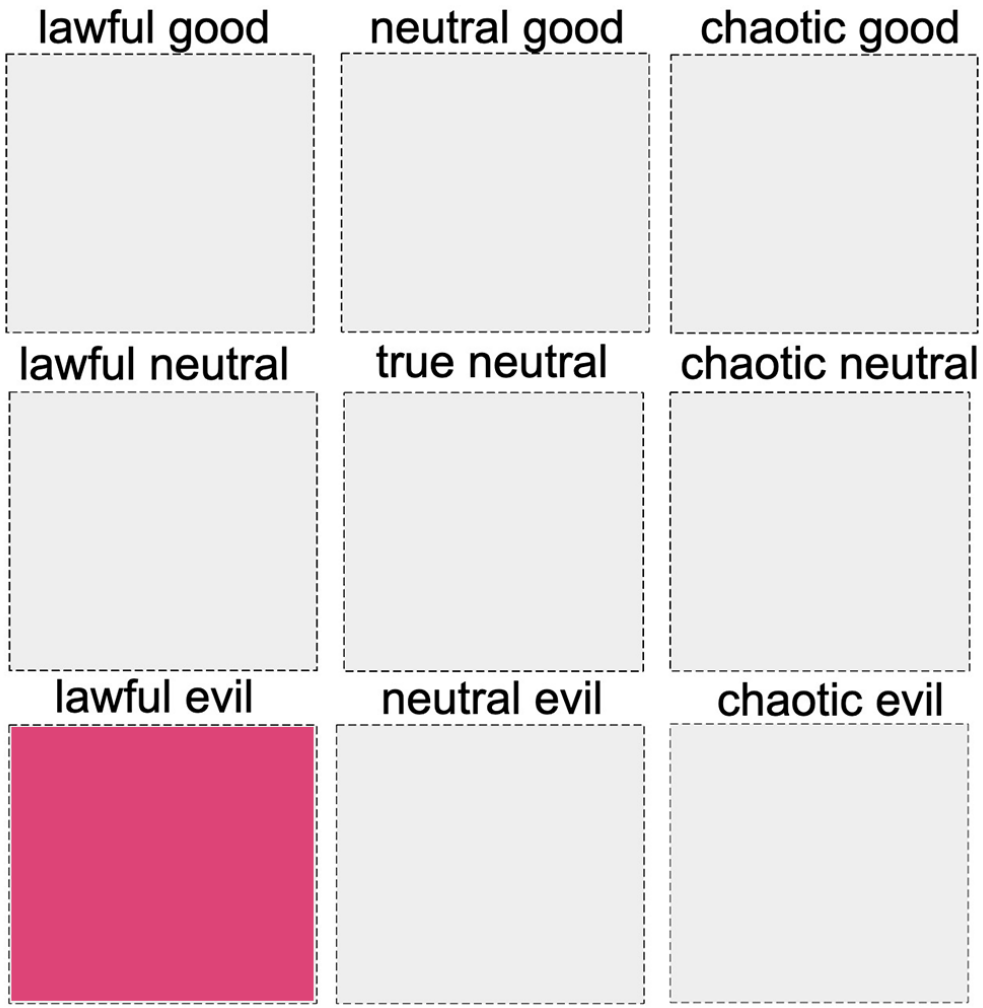
Important Data and Statistics

Where does Data Structures, as a subject, fall on this chart?



Important Data and Statistics

Where does Data Structures, as a subject, fall on this chart?



Important Data and Statistics

The course is going to be

Important Data and Statistics

The course is going to be

Course Structur

Timetable (subject to small changes)

classes	**314M**	**Tue, Thu 09:00–10:15(A), 10:30–11:45(B)**
office hours	**331W**	**Tue 12:15–14:15, Wed 11:45–12:45** **Thursday 12:15–13:15, or by appointment**
PSS		**Sat 14:00–16:00**
TA OH		**Mon 19:30–20:30, online**
Moodle key		**DataS-24a/DataS-24b**

Prerequisites

- CS120 Introduction to Object-Oriented Programming
- CS111 Discrete Mathematics

If unsure, talk to me right after this class!

Course Structure: OH and PSS

Any changes in the schedule are going to be reflected in the Timetable page.

OHs and PSS start from week II (Jan 24).

More information will be provided soon.

Assessment

Homework ($\times 5$)	10%	
Midterm I	20%	Monday, March 11, 18:00
Midterm II	20%	Monday, April 8, 18:00
Midterm III	20%	Tuesday, April 30, 18:00
Final Exam	30%	???

Homework Rules

Students are to submit their work **electronically** before the deadline.

The **format** of submitting homework assignments is posted as a PDF on Moodle.

Any programming involved in any of the homework assignments must be coded in **Java**.

Homework Rules

Late homework submissions are accepted with a penalty of **0.25%** per minute after the deadline.

How much penalty per hour?

Any collaboration or usage of materials (e.g. online sources) should be **explicitly acknowledged**. Acceptable for groups of **max. 2** people. The task is graded at **75%** of the actual score.

Any unacknowledged collaboration or usage of materials: the **whole** assignment graded **zero**

Usage of any AI tools (e.g. chatGPT) is prohibited.

Plagiarism Handling

Plagiarism is detected using a **software tool and human examination**.

After the potential plagiarism cases are identified, they are **reviewed** individually by the instructor.

If unacknowledged collaboration is detected, students will receive a grade of 0.

Students may appeal the charges during the instructor's offline OHs.

Grade Mapping

Grade	Grade Point	Percentile
A+	4	$[95, 100]$
A	4	$[90, 95)$
A−	3.7	$[85, 90)$
B+	3.3	$[80, 85)$
B	3	$[75, 80)$
B−	2.7	$[70, 75)$
C+	2.3	$[66, 70)$
C	2	$[62, 66)$
C−	1.7	$[58, 62)$
D+	1.3	$[55, 58)$
D	1	$[53, 55)$
D−	0.7	$[50, 53)$
F	0	$[0, 50)$

How to Succeed in This Course?

- Work regularly!
  - The course is intensive and incremental, with new topics introduced at each class.
  - Try to go over the topics covered in class later the same day and make sure that you understand every concept discussed.

- Do the reading!
  - Classes cannot cover all the details. Reading the materials ensures better understanding of the topics.
  - On average you will have 35 pages of reading per week. Don't let it pile up!
  - Simultaneously read the slides and the textbook.
    - Read a page from the slides.
    - Read the corresponding paragraph/section from the book.
    - Re-read the slide page and move forward.

How to Succeed in This Course? Cont.

- Concentrate on thinking!
  - You may not be given typical problems during exams/quizzes.
  - Problems often should be solved by thinking and applying your theoretical knowledge.
  - Enhance your understanding of covered concepts to solve new types of problems.

- Practice!
  - Mastery comes with experience. Solving problems using the new algorithms enhances understanding of the material.
  - The more you do coding the more speed and accuracy you develop.

How to Succeed in This Course? Cont.

Make good use of the available resources which include

- Regular lectures
- PSS
- OHs
- Textbook (highly recommended, 10/10 would read again)

How to Succeed in This Course? Cont.

Make good use of the available resources which include

- Regular lectures
- PSS
- OHs
- Textbook (highly recommended, 10/10 would read again)
- Geeks for Geeks

How to Succeed in This Course? Cont.

Make good use of the available resources which include

- Regular lectures
- PSS
- OHs
- Textbook (highly recommended, 10/10 would read again)
- Geeks for Geeks
- Indian youtubers (bless their hearts)

How to Succeed in This Course? Cont.

Make good use of the available resources which include

- Regular lectures
- PSS
- OHs
- Textbook (highly recommended, 10/10 would read again)
- Geeks for Geeks
- Indian youtubers (bless their hearts)
- Stack Exchange/Overflow

How to Succeed in This Course? Cont.

Make good use of the available resources which include

- Regular lectures
- PSS
- OHs
- Textbook (highly recommended, 10/10 would read again)
- Geeks for Geeks
- Indian youtubers (bless their hearts)
- Stack Exchange/Overflow
- ChatGPT or other AI tools

How to Succeed in This Course? Cont.

Make good use of the available resources which include

- Regular lectures
- PSS
- OHs
- Textbook (highly recommended, 10/10 would read again)
- Geeks for Geeks
- Indian youtubers (bless their hearts)
- Stack Exchange/Overflow
- ChatGPT or other AI tools
- Google

How to Succeed in This Course? Cont.

Make good use of the available resources which include

- Regular lectures
- PSS
- OHs
- Textbook (highly recommended, 10/10 would read again)
- Geeks for Geeks
- Indian youtubers (bless their hearts)
- Stack Exchange/Overflow
- ChatGPT or other AI tools
- Google
- Syllabus

How to Succeed in This Course? Cont.

- Know your grade!
  - Right now you grade is a **100**!
  - You don't get points throughout the course, you lose them by skipping assignments and making mistakes.
  - Subtract the points lost from the 100 to keep track of your grade.
  - Choose the grade you want to have, and aim one step higher. **Ex.** If you want an A in this course, aim to stay in the A+ range. If you don't succeed, chances are high that you will end up in the A range anyway.
  - **Your grade will be calculated in this way on Moodle**.
- Do all the assignments!
  - If you skip assignments, problems will arise later in the course.
  - Each skipped assignment equals to 2 lost points.
  - Solving homework tasks will better prepare you for the exams.

How to Succeed in This Course? Cont.

DO NOT CHEAT

- Copying == Disrespecting
  You disrespect the person you copy from, your classmates that work harder than you, your TAs, Instructors and the University
- Channel your anger in the right direction
  If you let someone copy your work and are caught on an unacknowledged collaboration, be angry at yourself and that person, not the TAs and Instructors.
- You should always come first!
  Never be ashamed of rejecting classmates' requests if you find them hurtful for your grade.
- Offer help if you have time
  Ask a person if they have particular questions about the homework and help them if you so wish.

How to Succeed in This Course? Cont.

DO NOT

- use your phone in class.
  - Paying attention to something for 75 minutes is indeed painful.
  - However, it is much better to spend this time engaged in class than re-spend twice or thrice as much outside of it trying to catch up.
  - Remember your tuition fee (**1.700.000 AMD**)

- skip topics even if you think you know them well.
  - There is always place for improvement and further practice
  - If you don't get into the right pace from the beginning of the course, it may become exponentially difficult to do so later on

How to Succeed in This Course? Cont.

Read the document "Advice For Future Generations" to get some feedback from the students that took this course before you.

The following questions are answered there:

- *What learning habits of yours worked for you in the scope of this course?*
- *What learning habits of yours failed in the scope of this course?*
- *If you had the opportunity to start this course from the beginning, would you change the way you have studied? If yes, how?*
- *What advice would you give to your past self and the future students who will take this course?*

Syllabus and Moodle

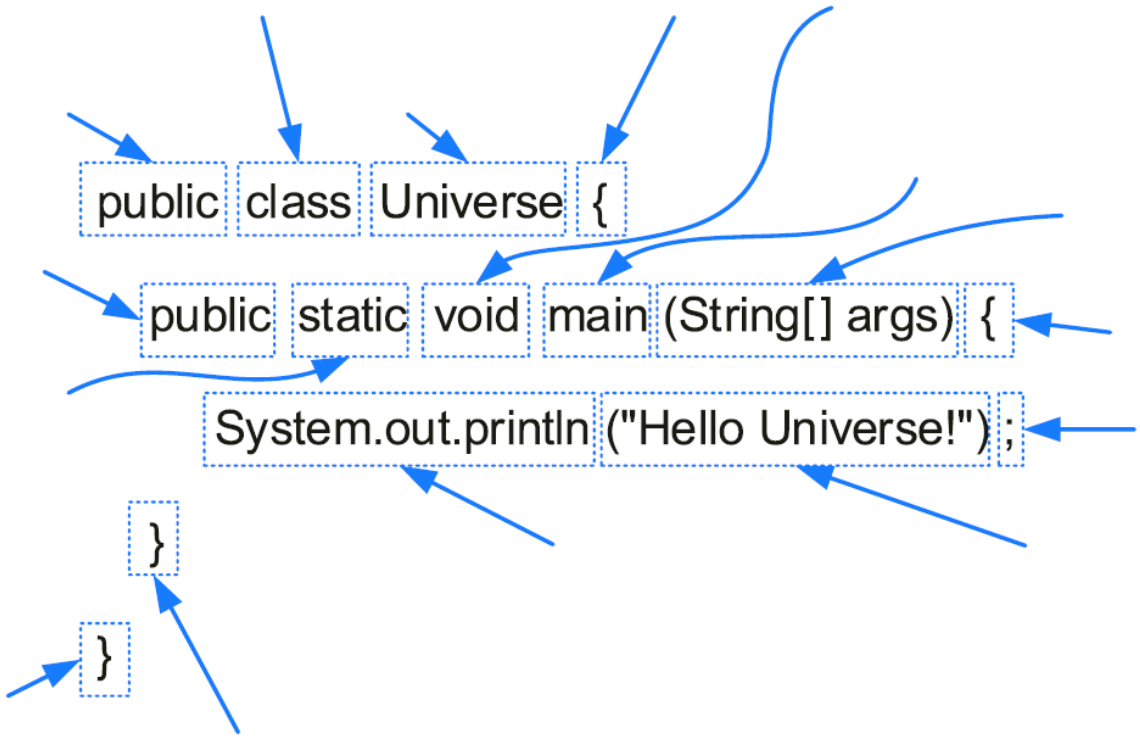
The syllabus and other materials will be available on Moodle

Enroll on Moodle with the key provided today ()

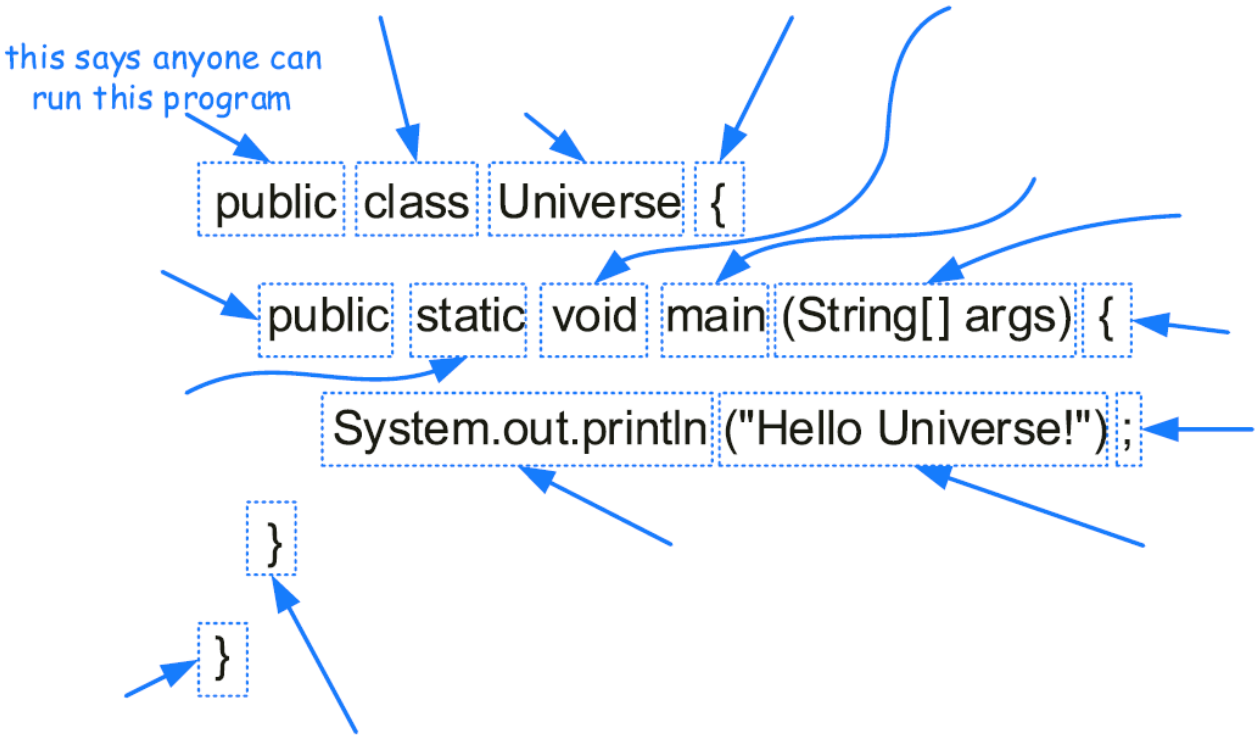
The course is divided into three main parts:

- complexity, recursion, search, and sorting ($\sim$ 3 weeks)
- linear data structures ($\sim$ 4 weeks)
- non-linear data structures ($\sim$ 8 weeks)

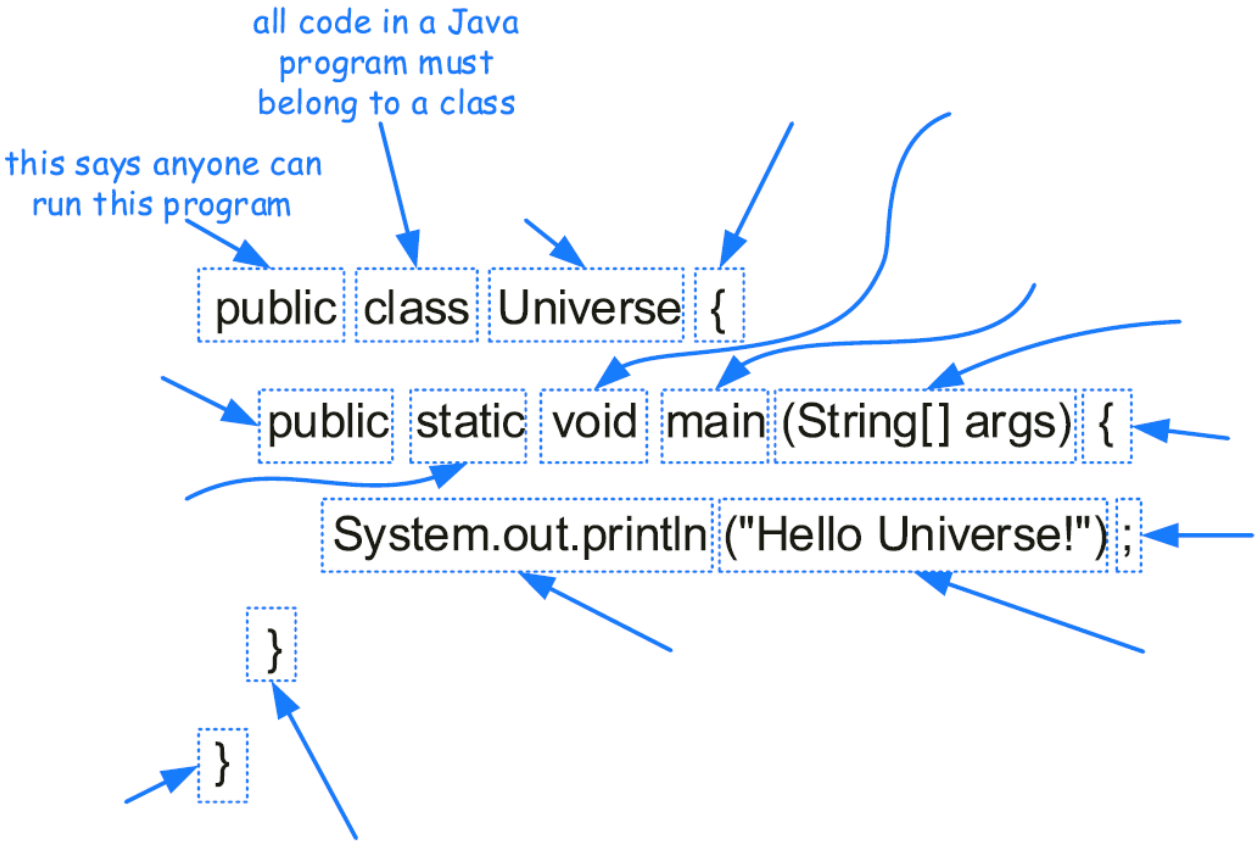
Simple Program in Java



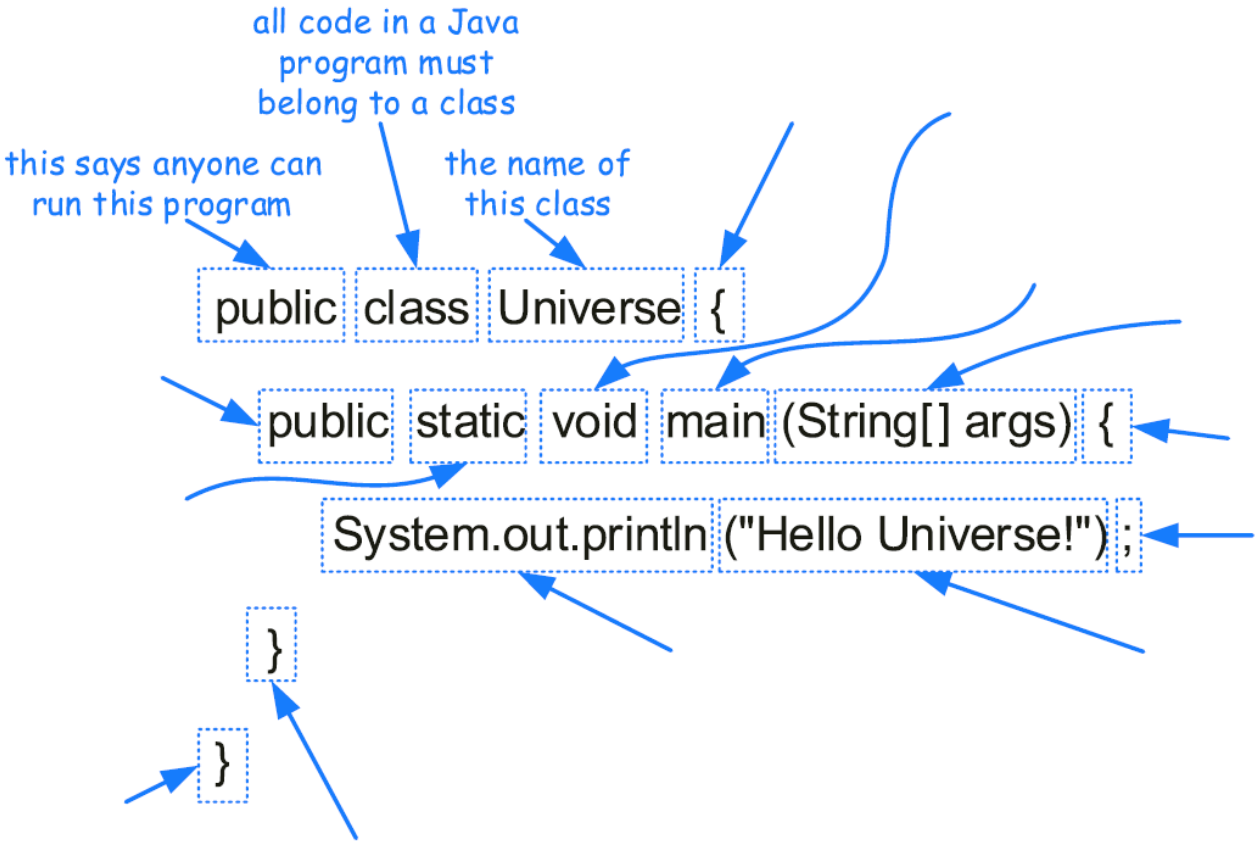
Simple Program in Java



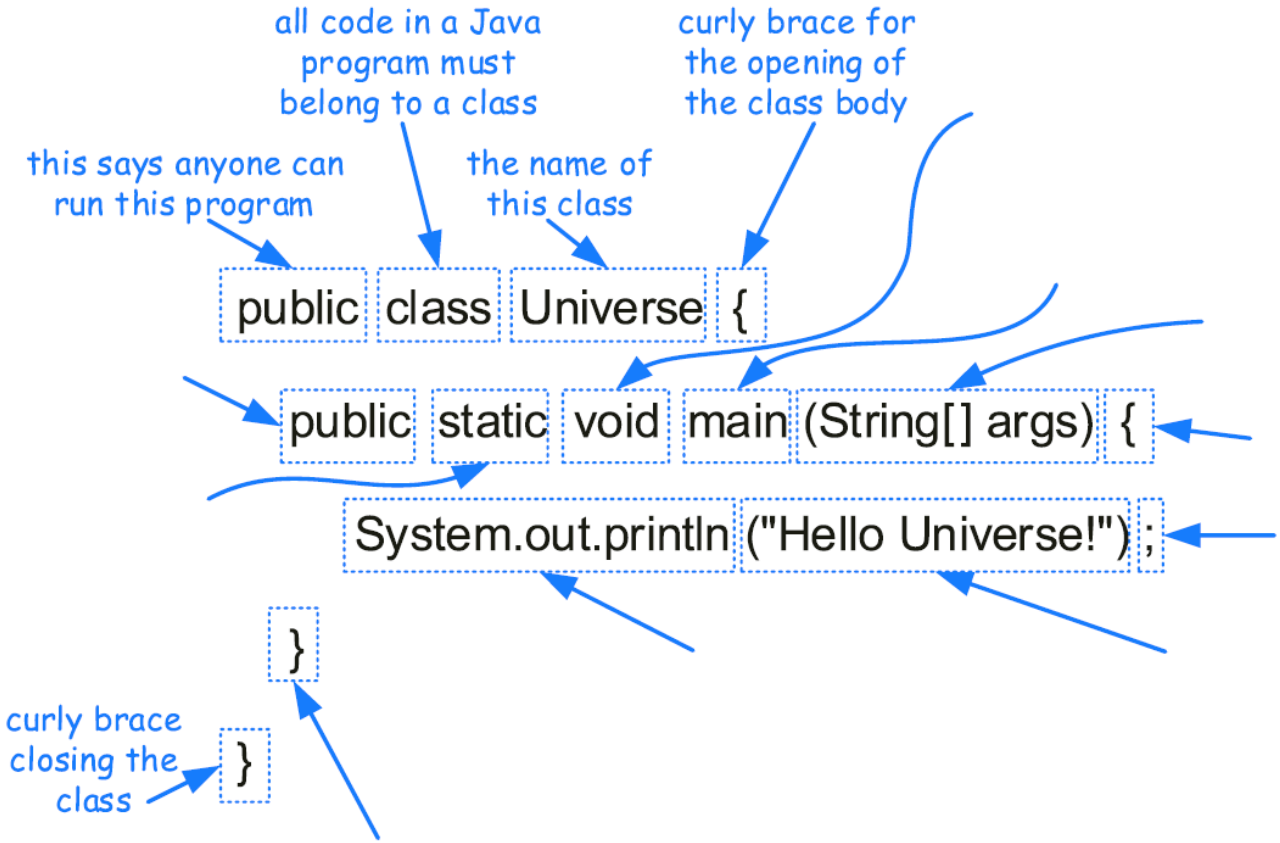
Simple Program in Java



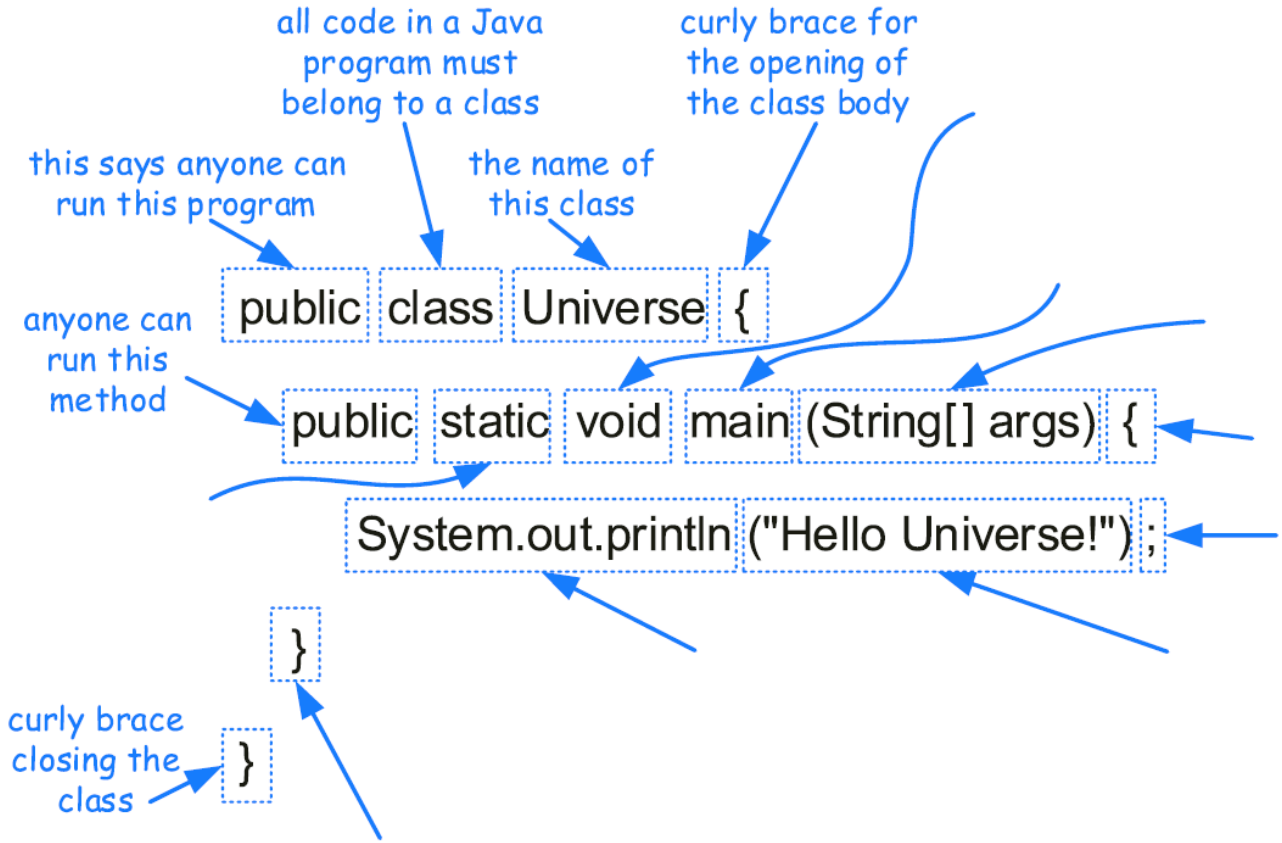
Simple Program in Java



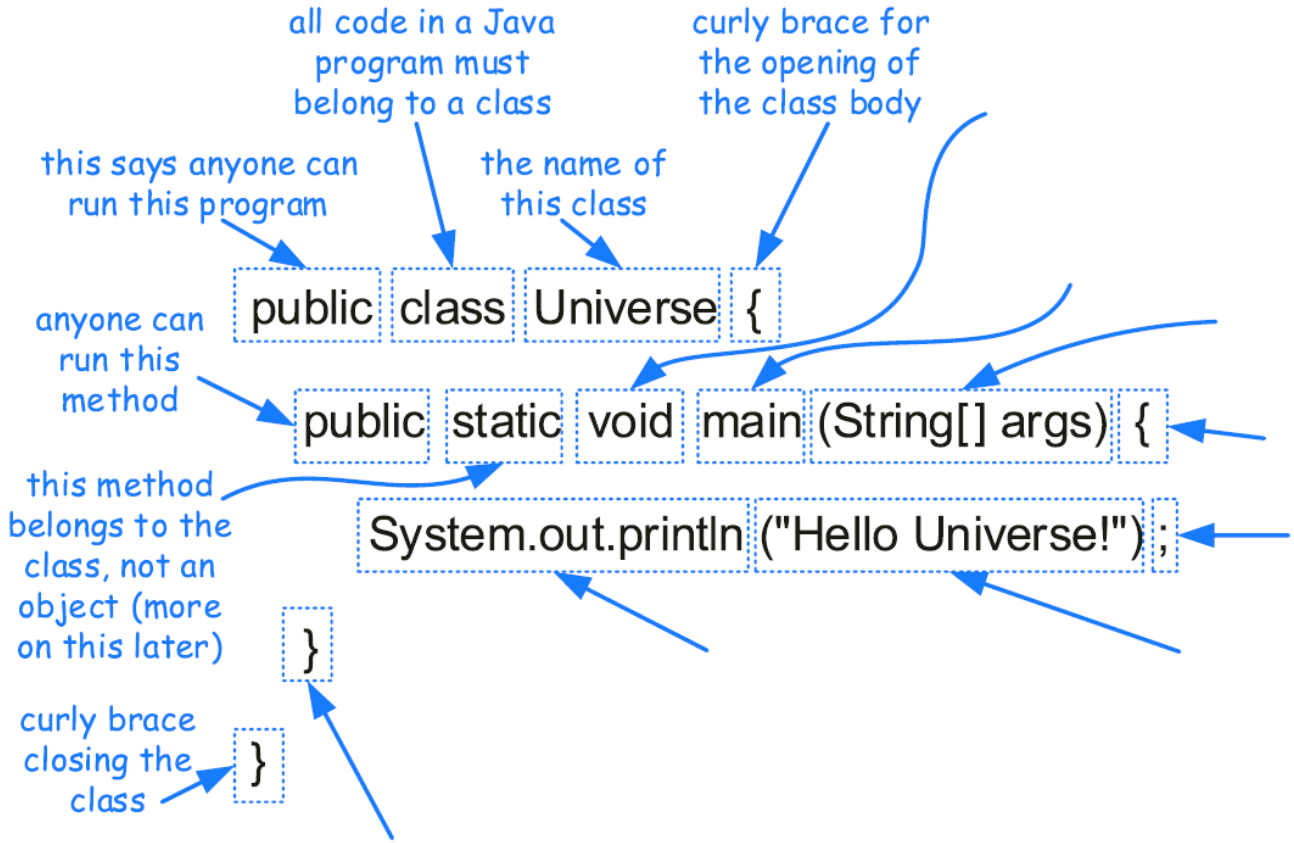
Simple Program in Java



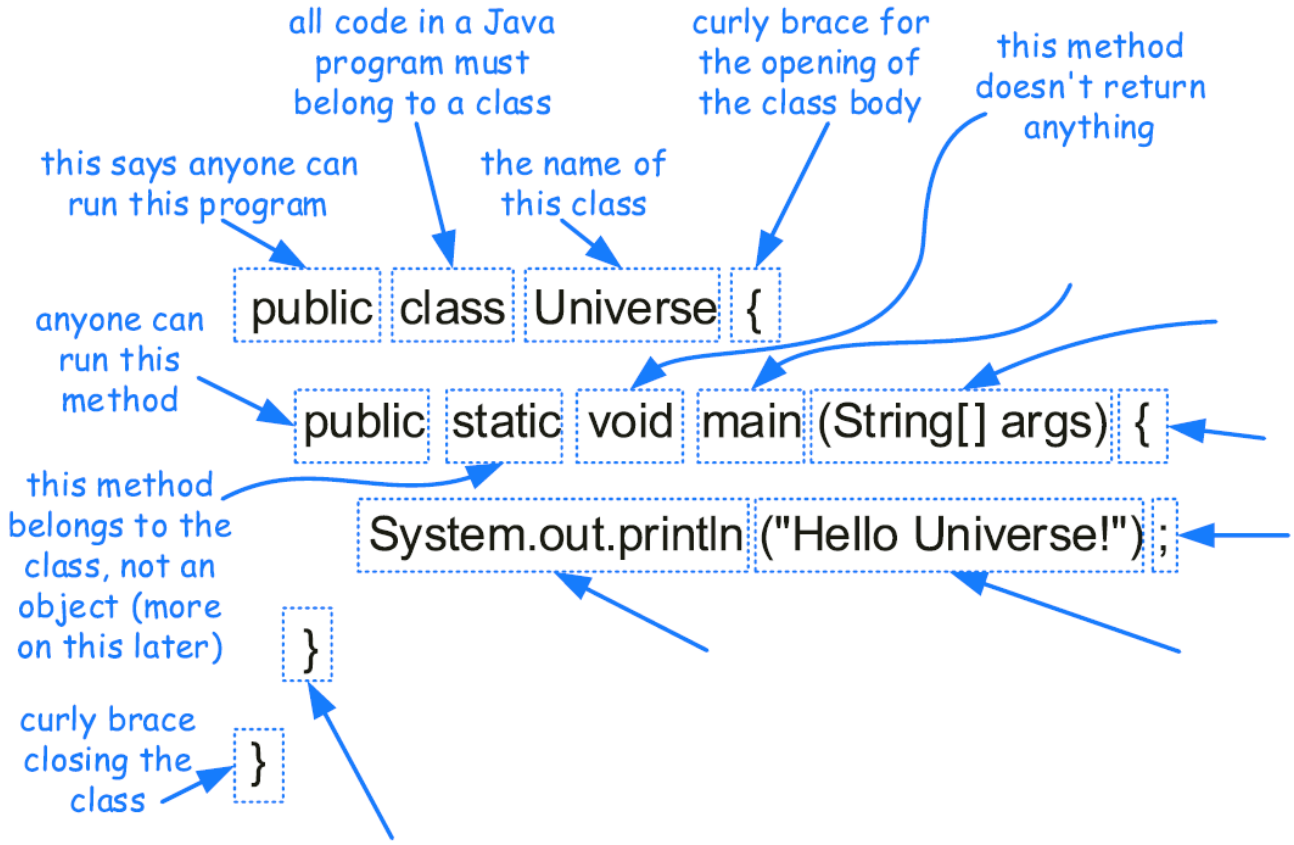
Simple Program in Java



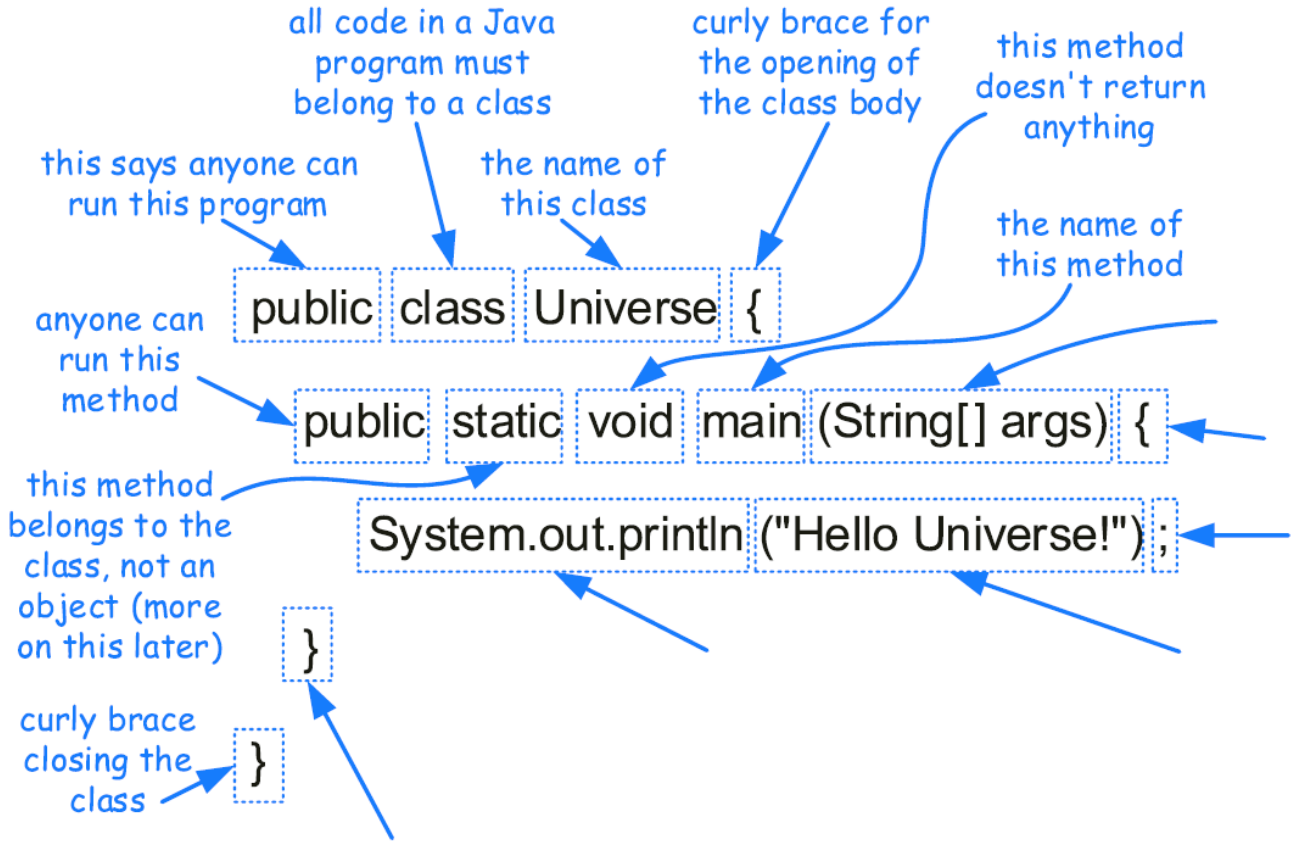
Simple Program in Java



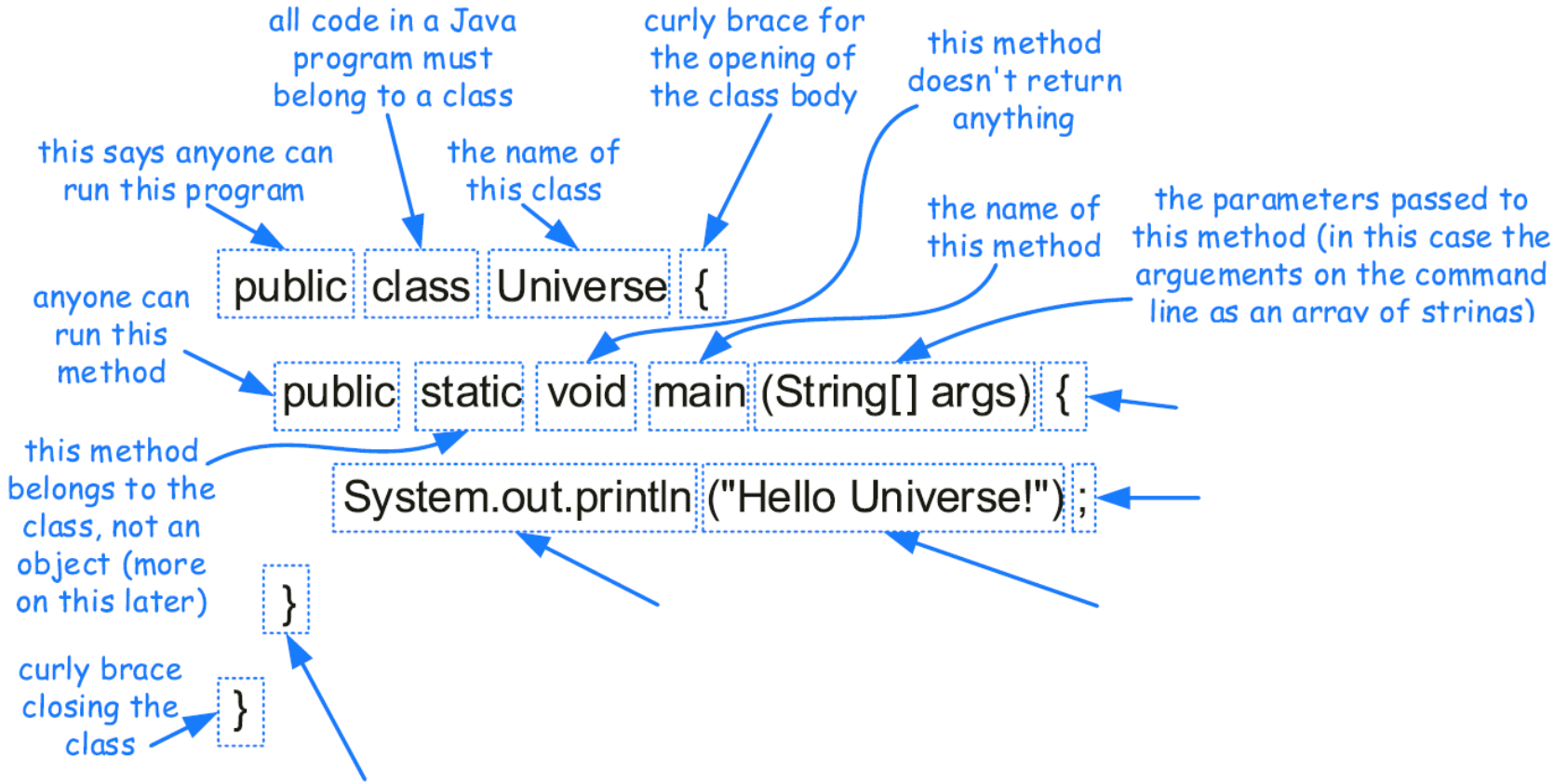
Simple Program in Java



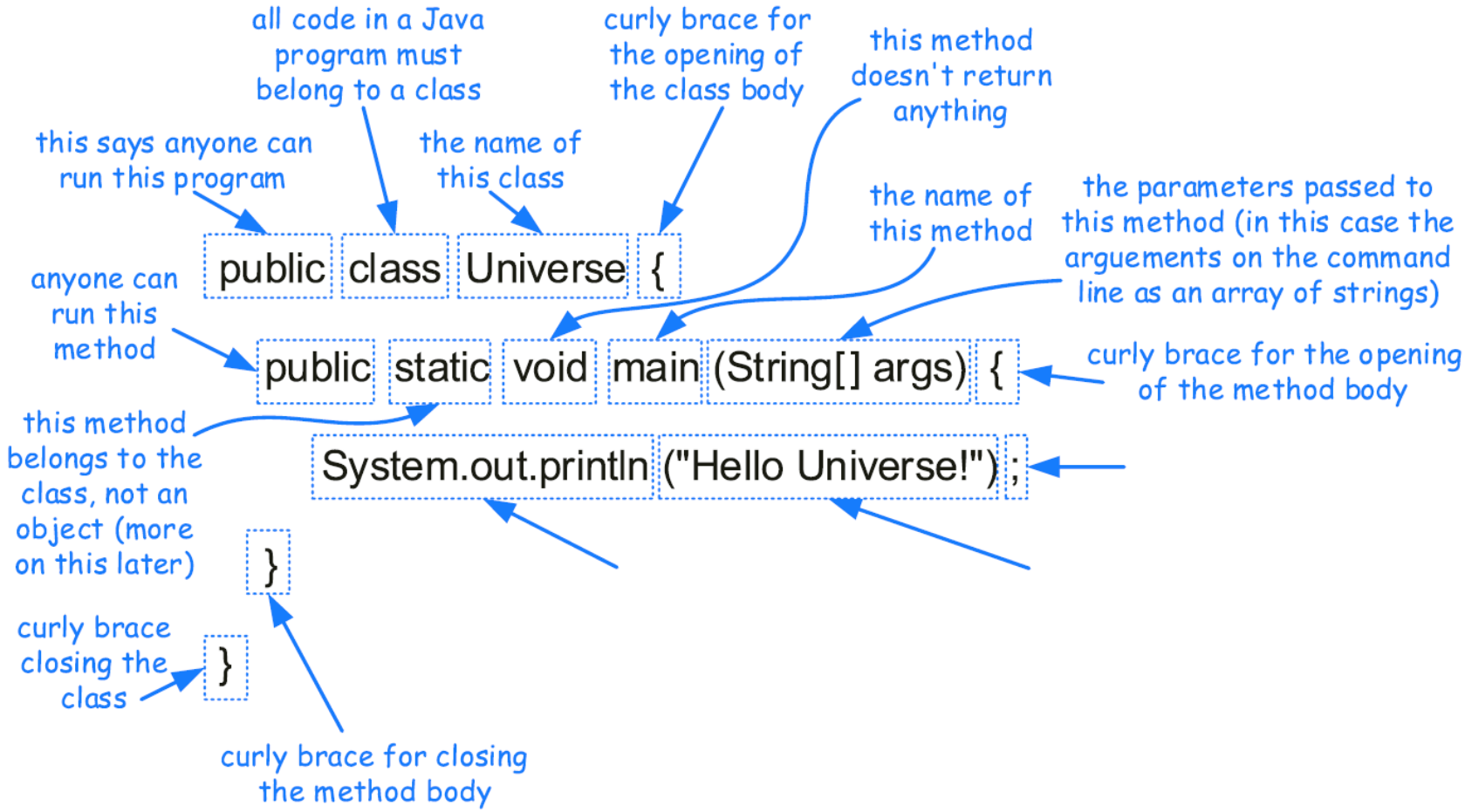
Simple Program in Java



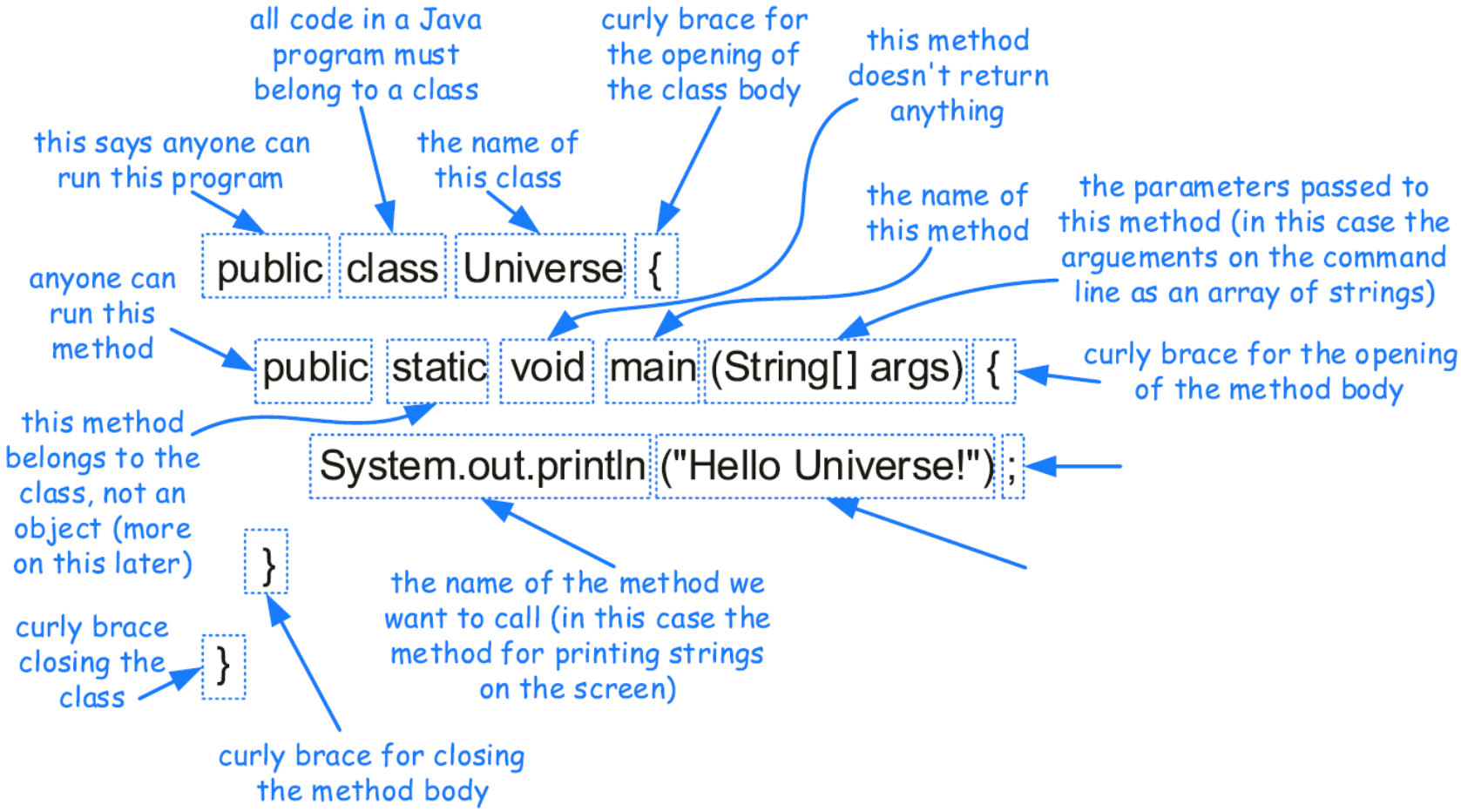
Simple Program in Java



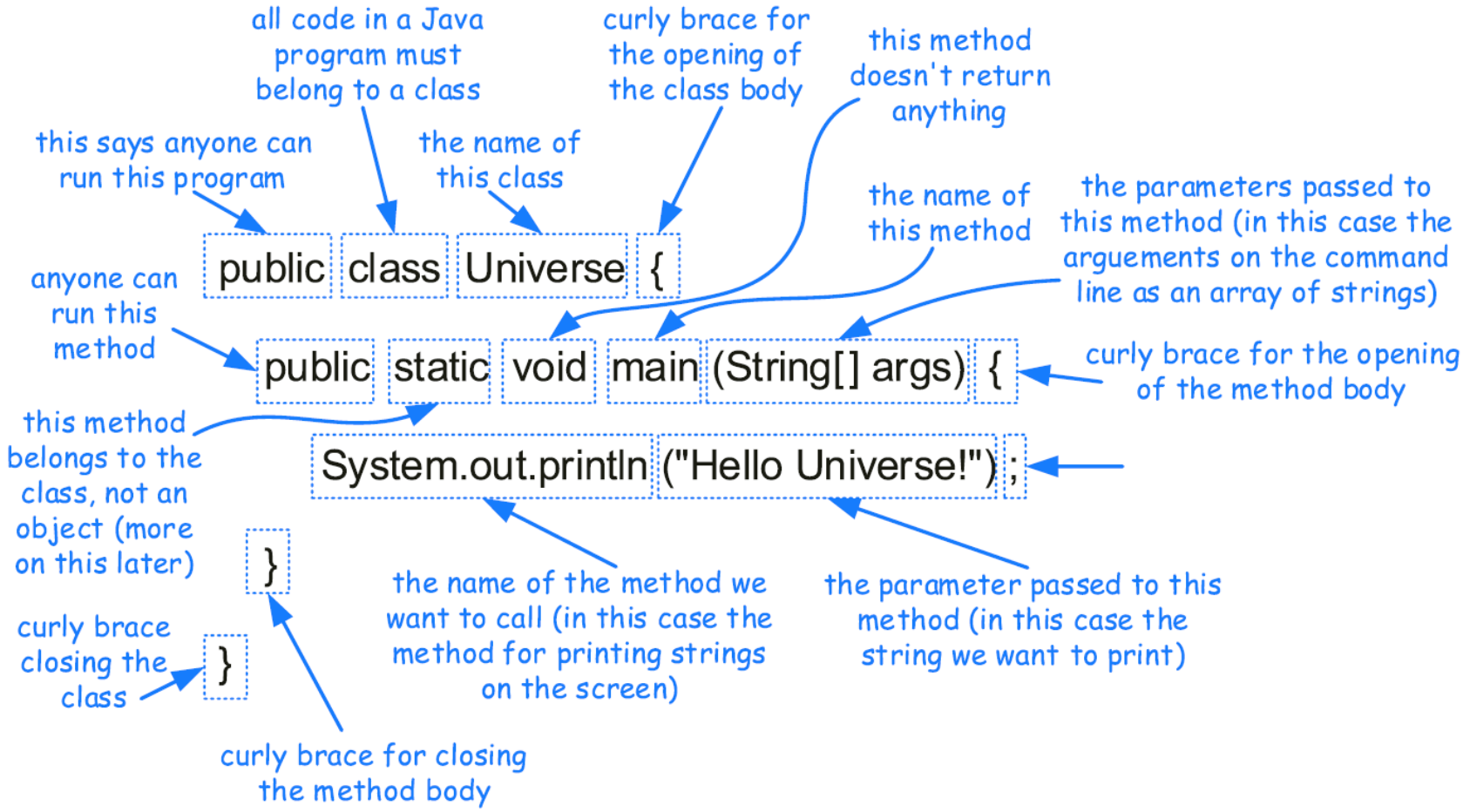
Simple Program in Java



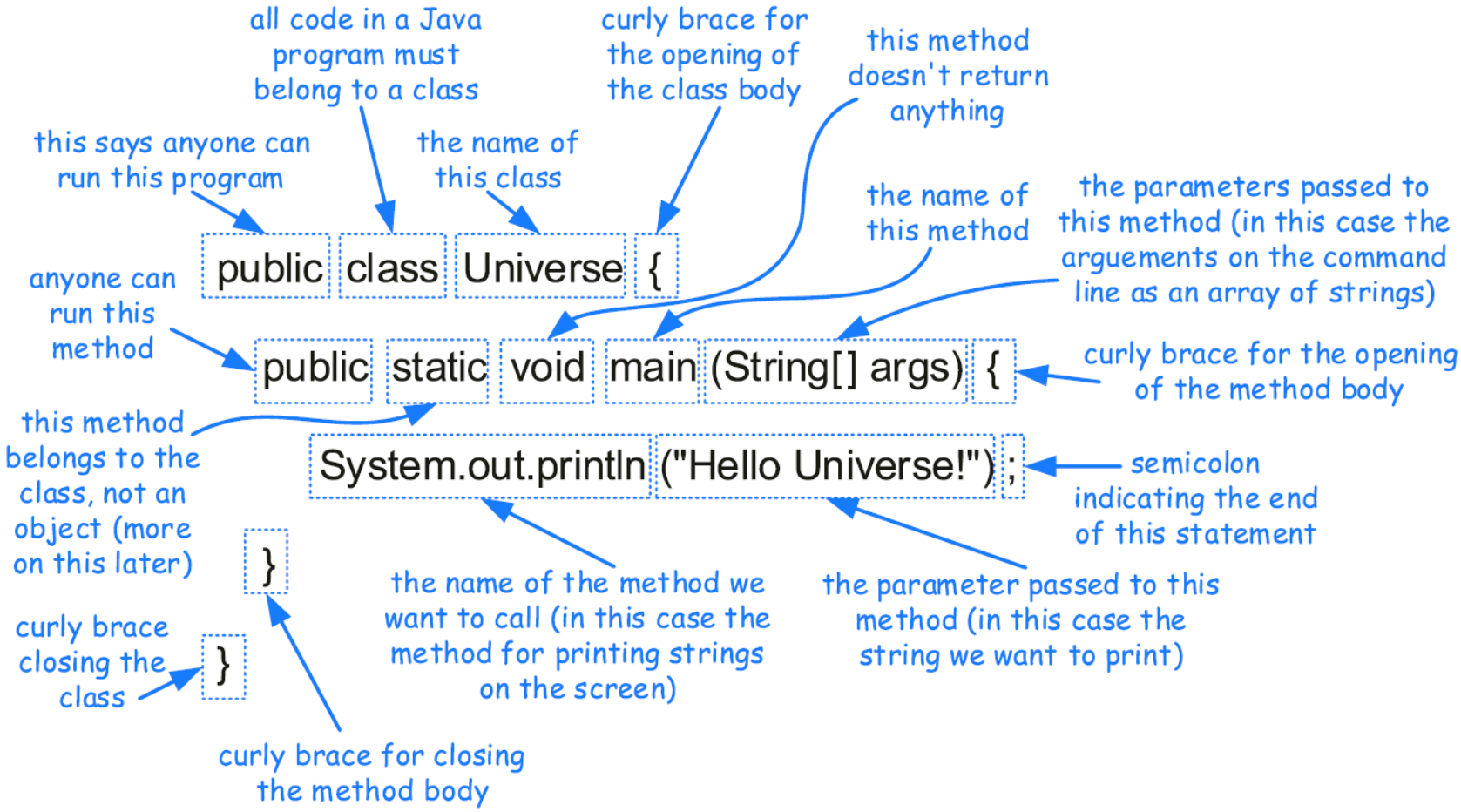
Simple Program in Java



Simple Program in Java



Simple Program in Java



Base Types

Programming languages typically have several base types, which are basic ways of storing data

A variable can be declared to hold any base type and it can later be reassigned to hold another value of the same type

boolean	a boolean value: true or false
char	16-bit Unicode character
byte	8-bit signed two's complement integer
short	16-bit signed two's complement integer
int	32-bit signed two's complement integer
long	64-bit signed two's complement integer
float	32-bit floating-point number (IEEE 754-1985)
double	64-bit floating-point number (IEEE 754-1985)

```
boolean flag = true;
boolean verbose, debug;
char grade = 'A';
byte b = 12;
short s = 24;
int i, j, k = 257;
long l = 890L;
float pi = 3.1416F;
double e = 2.71828, a = 6.022e23;
```

Class Types

Every **object** is an instance of a **class**, which serves as the type of the object and as a blueprint (scheme), defining the

- **data** which the object stores (**instance variables, or fields**)
- **methods** for accessing and modifying that data.

```
public class Counter {
  private int count;                                  // a simple integer instance variable
  public Counter( ) { }                               // default constructor (count is 0)
  public Counter(int initial) { count = initial; }          // an alternate constructor
  public int getCount( ) { return count; }                    // an accessor method
  public void increment( ) { count++; }                        // an update method
  public void increment(int delta) { count += delta; }         // an update method
  public void reset( ) { count = 0; }                          // an update method
}
```

Class Example

```
public class CounterDemo {
  public static void main(String[ ] args) {
    Counter c;                          // declares a variable; no counter yet constructed
    c = new Counter( );                 // constructs a counter; assigns its reference to c
    c.increment( );                     // increases its value by one
    c.increment(3);                     // increases its value by three more
    int temp = c.getCount( );           // will be 4
    c.reset( );                         // value becomes 0
    Counter d = new Counter(5);         // declares and constructs a counter having value 5
    d.increment( );                     // value becomes 6
    Counter e = d;                      // assigns e to reference the same object as d
    temp = e.getCount( );               // will be 6 (as e and d reference the same counter)
    e.increment(2);                     // value of e (also known as d) becomes 8
  }
}
```

Data Structures

A **data structure** is a particular way of organizing data in a computer so that it can be used effectively.

A **data structure** is a

- data organization,
- management and
- storage format

that enables efficient access and modification

It is a *collection* of data values, the *relationships* among them, and the *functions or operations* that can be applied to the data

Simple example: arrays

Abstract Data Types

Abstraction breaks down a system to its most fundamental parts. An abstract class or an interface is like a *blueprint* of a concrete class.

Applying the abstraction paradigm to the design of data structures gives rise to **abstract data types (ADTs)**

An ADT specifies the **type** of data stored, the supported **operations**, and the **types of parameters** of the operations

An ADT specifies *what* each operation does, but not *how* it does it

The collective set of behaviours supported by an ADT is its **public interface**

Sample Program in Java: 1

```
1  public class CreditCard {
2    // Instance variables:
3    private String customer;          // name of the customer (e.g., "John Bowman")
4    private String bank;              // name of the bank (e.g., "California Savings")
5    private String account;           // account identifier (e.g., "5391 0375 9387 5309")
6    private int limit;                // credit limit (measured in dollars)
7    protected double balance;         // current balance (measured in dollars)
8    // Constructors:
9    public CreditCard(String cust, String bk, String acnt, int lim, double initialBal) {
10     customer = cust;
11     bank = bk;
12     account = acnt;
13     limit = lim;
14     balance = initialBal;
15   }
16   public CreditCard(String cust, String bk, String acnt, int lim) {
17     this(cust, bk, acnt, lim, 0.0);                  // use a balance of zero as default
18   }
```

Sample Program in Java: 2

```
19    // Accessor methods:
20    public String getCustomer( ) { return customer; }
21    public String getBank( ) { return bank; }
22    public String getAccount( ) { return account; }
23    public int getLimit( ) { return limit; }
24    public double getBalance( ) { return balance; }
25    // Update methods:
26    public boolean charge(double price) {          // make a charge
27      if (price + balance > limit)                 // if charge would surpass limit
28        return false;                              // refuse the charge
29      // at this point, the charge is successful
30      balance += price;                            // update the balance
31      return true;                                 // announce the good news
32    }
33    public void makePayment(double amount) {       // make a payment
34      balance -= amount;
35    }
36    // Utility method to print a card's information
37    public static void printSummary(CreditCard card) {
38      System.out.println("Customer = " + card.customer);
39      System.out.println("Bank = " + card.bank);
40      System.out.println("Account = " + card.account);
41      System.out.println("Balance = " + card.balance);   // implicit cast
42      System.out.println("Limit = " + card.limit);       // implicit cast
43    }
44    // main method shown on next page...
45  }
```

Sample Program in Java: 3

```
1   public static void main(String[ ] args) {
2     CreditCard[ ] wallet = new CreditCard[3];
3     wallet[0] = new CreditCard("John Bowman", "California Savings",
4                                "5391 0375 9387 5309", 5000);
5     wallet[1] = new CreditCard("John Bowman", "California Federal",
6                                "3485 0399 3395 1954", 3500);
7     wallet[2] = new CreditCard("John Bowman", "California Finance",
8                                "5391 0375 9387 5309", 2500, 300);
9
10    for (int val = 1; val <= 16; val++) {
11      wallet[0].charge(3*val);
12      wallet[1].charge(2*val);
13      wallet[2].charge(val);
14    }
15
16    for (CreditCard card : wallet) {
17      CreditCard.printSummary(card);          // calling static method
18      while (card.getBalance( ) > 200.0) {
19        card.makePayment(200);
20        System.out.println("New balance = " + card.getBalance( ));
21      }
22    }
23  }
```

Summary

Reading

Java language review: Chapter 1 Java Primer

OOP review: Chapter 2 Object-Oriented Design

Questions?

Acknowledgement: Some contents presented today and in future classes are based on Varduhi Yeghiazaryan's course slides at AUA.