CMPE 101
Object Oriented Programming

İstanbul
Bilgi University

Dr. Emel Küpçü

Department of Computer Engineering

İstanbul Bilgi University

Week-1: Introduction to OOP Concept and the Java Language, Console Input & Output Operations

CMPE 101 Object Oriented Programming

Course Description:

- This course introduces the fundamentals of Object-Oriented Programming (OOP) using Java, covering core concepts such as classes, methods, encapsulation, inheritance, and polymorphism.
- Students will develop problem-solving skills through hands-on exercises involving console I/O, conditional statements, loops, and arrays.
- Advanced topics include exception handling, file I/O, generic collections, and graphical user interfaces (GUI) to build real-world applications.
- The course also explores UML diagrams for software design and emphasizes best coding practices.
- By the end of the course, students will be proficient in writing efficient, modular, and maintainable Java programs.

Textbook:

- ISBN-9781292223858 Java : how to program early objects / Paul Deitel, Deitel & Associates, Inc., Harvey Deitel, Deitel & Associates, Inc., 2018, The 11th edition, Pearson.

Course Syllabus

- 1. Week: Introduction to OOP Concept and the Java Language, Input & Output Operations
- 2. Week: Methods, Constructors, Parameters, Primitive Data Types and Encapsulation
- 3. Week: Conditional Statements
- 4. Week: Looping Statements, Loop Control and Logical Operators
- 5. Week: Static Variables, Static Methods, Method Overloading
- 6. Week: Programming with Arrays
- 7. Week: Midterm

Course Syllabus (Cont.)

- 8. Week: Building Classes, Enum Types
- 9. Week: Programming with Inheritance
- 10. Week: Polymorphism, Abstract Classes and Interfaces
- 11. Week: Exception Handling
- 12. Week: File Input/Output
- 13. Week: Generic Classes and Methods
- 14. Week: UML diagrams
- 15. Week: Graphical User Interfaces
- 16. Week Final
- 17. Week Final

Course Syllabus (Cont.)

	Quantity	Percentage
Assignments	3	15
Problem Solving (Labs)	10	10
Quiz	1 (avg.of N)	10
Midterm(s)	1	25
Final exam	1	40

Course Syllabus (Cont.)

Students are required to attend all courses, practical sessions, laboratory studies, and all exams and other academic studies.

Attendance is mandatory and will also **impact your quiz grade**.

- There will be **N quizzes**, though the exact number is not predetermined. Quizzes may be either **announced in advance** or **given randomly**.
- Students who attend at least **70% of the lectures** will have their lowest quiz score excluded from the final quiz average.
  - Let **N** be **the total number of quizzes** at the end of the semester.
  - If attendance percentage ≥ **70%**:

    Exclude the lowest quiz score.

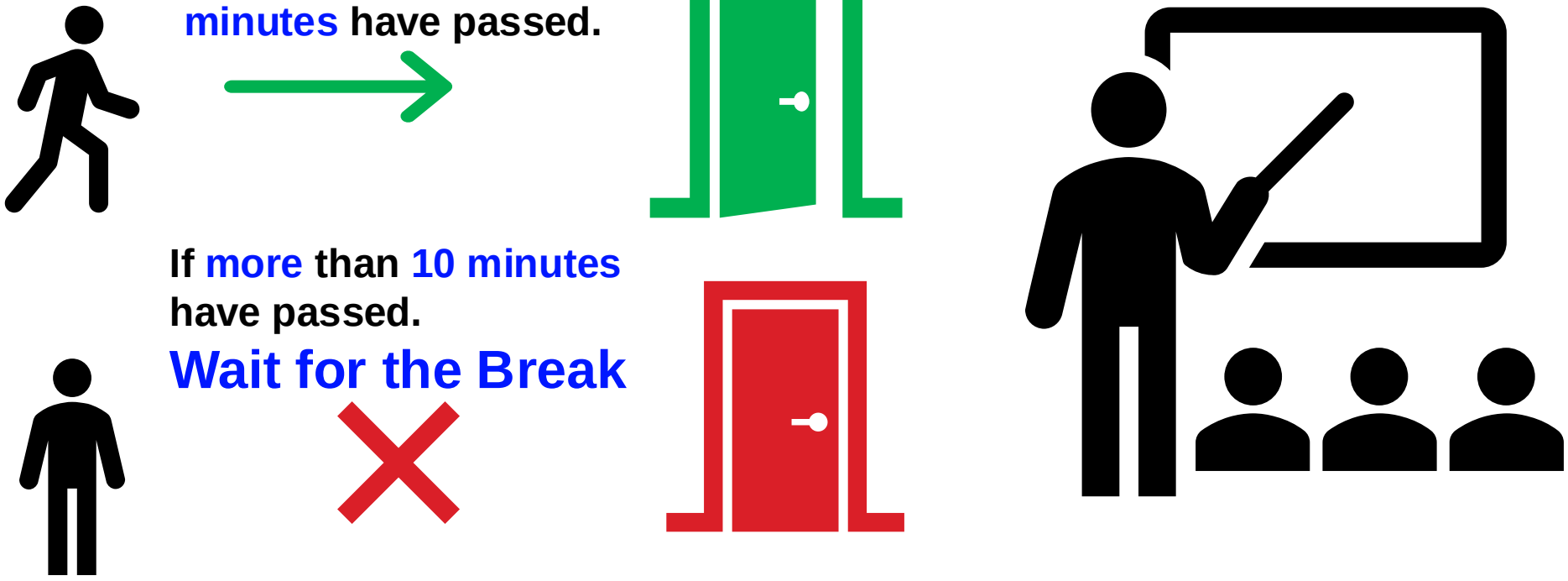
    Compute the **average** using the **top** (**N-1**) **quiz scores**.
  - Else:

    Compute the **average** using **all N quiz scores**.

Arrive On Time for Class!!

- Late arrivals can be disruptive,
- Being on time demonstrates respect for both your classmates and your education.

Let's create a focused and productive environment together!



This is only valid during <u>the first hour of the classes</u>.

Turn off Your Devices

- Using devices is not allowed during the lecture's learning phase.

. . .

Object Oriented Programming

Key Features:

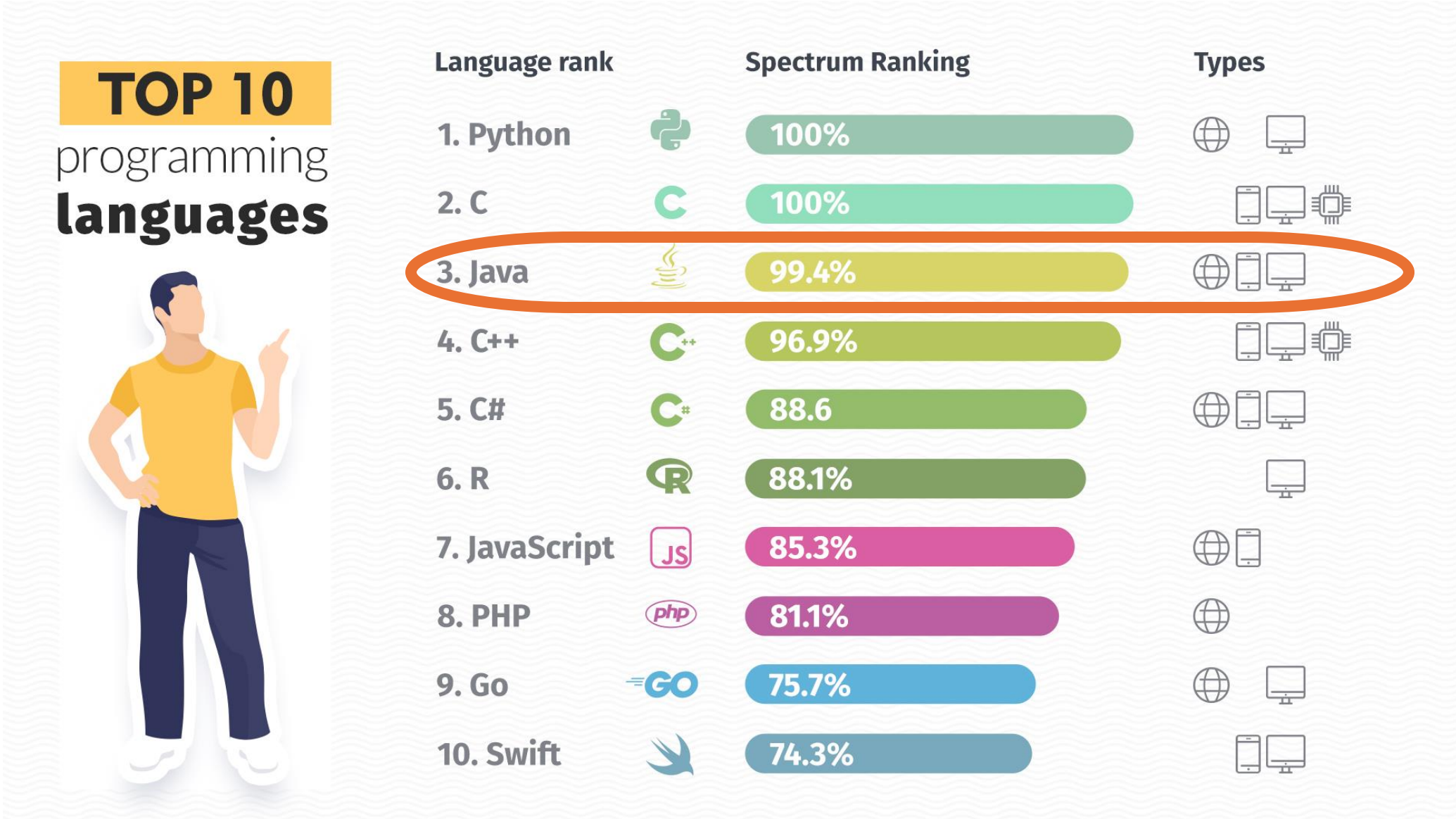
- **Inheritance**
  - Allows new classes to inherit properties and methods from existing ones.
    - If you have a class for "Animal," you can create a "Dog" class that automatically gets the basic animal features and then adds specific features for dogs.
- **Encapsulation:**
  - Groups data and functions together while limiting direct access to some of the object's components.
    - Some important parts are hidden and can't be changed directly, protecting the data from unwanted changes.
- **Polymorphism**
  - The same function name can be used for different types of objects, but each object can have its own specific behavior for that function.
    - Ex: If you have a class “Animal” and different subclasses like “Dog” and “Cat”, both can have a method called speak(). Even though the method has the same name, a dog may bark while a cat may meow.
- **Abstraction**
  - Hides the internal workings of objects, exposing only essential components.
    - Ex, when you drive a car, you only need to know how to use the steering wheel, pedals, and gear shift—this is the **essential interface**. You don’t need to know how the engine works, how fuel is converted to energy, or how the brake system functions—these are the **hidden internal workings**.

Object Oriented Programming (cont.)

Key Benefits:

- **Code Reusability**: Inheritance allows the reuse of existing code.
- **Data Security**: Encapsulation restricts direct access to data.
- **Modularity**: Encapsulation divides the program into logical units.
- **Flexibility**: Polymorphism allows methods to work with different types of objects.

Programming Languages



https://www.slideshare.net/slideshow/programming-languages-java-python-sql-and-compare-between-c-and-python/120994721?from_search=0

Java

- Java is one of the powerful general-purpose programming languages, created in 1995 by Sun Microsystems (now owned by Oracle).
- Java is Object-Oriented.
- Java code is always written in the form of classes and objects.
- References:
  - The Java Tutorials by Oracle
    - https://docs.oracle.com/javase/tutorial/java/index.html
  - An Introduction to Object-Oriented Programming with Java, 5th Edition
    - https://lecture-notes.tiu.edu.iq/wp-content/uploads/2021/10/An-Introduction-to-Object-Oriented-Programming-with-Java-by-C.-Thomas-Wu-z-lib.org_.pdf

Devices		
Access control systems	Airplane systems	ATMs
Automobiles	Blu-ray Disc™ players	Building controls
Cable boxes	Copiers	Credit cards
CT scanners	Desktop computers	e-Readers
Game consoles	GPS navigation systems	Home appliances
Home security systems	Internet-of-Things gateways	Light switches
Logic controllers	Lottery systems	Medical devices
Mobile phones	MRIs	Network switches
Optical sensors	Parking meters	Personal computers
Point-of-sale terminals	Printers	Robots
Routers	Servers	Smart cards
Smart meters	Smartpens	Smartphones
Tablets	Televisions	Thermostats
Transportation passes	TV set-top boxes	Vehicle diagnostic systems

Fig. 1.1 | Some devices that use Java.

Top Companies

- **Google**
  - Java is essential at Google, **powering Android** (the most popular mobile OS) and **many internal systems** due to its scalability and reliability.
- **Amazon**
  - Amazon uses Java across its **e-commerce platform and AWS**, allowing it to **handle millions of transactions and scale for vast numbers of users**.
- **Netflix**
  - Java supports Netflix's complex backend architecture, **ensuring performance, stability, and scalability for high-volume streaming**.
- **Uber**
  - Uber relies on Java **for real-time operations** like driver-rider matching, payments, and logistics, benefiting from **Java's efficiency and multi-threading**.
- **LinkedIn**
  - LinkedIn uses Java **to manage user data**, **connections**, and **interactions**, enabling smooth scalability for its large network.

General Uses of Java

- Java is the primary language for **Android development**, and many Android apps are built using Java.
  - Android's API and SDK are primarily based on Java
- Java is used for building scalable, high-performance **web applications**, especially for enterprise-level solutions.
  - Java-based frameworks like Spring, Hibernate, and Struts are commonly used to develop robust backends for web applications.
- Java is highly favored for **enterprise-level applications** due to its scalability, security, and reliability.
  - Large companies and government agencies use Java to build software systems for customer relationship management (CRM), enterprise resource planning (ERP), and financial services.
- Java is used for scientific and mathematical applications that require high precision and stability.
  - Simulation software, data analysis tools, and scientific calculations.

General Uses of Java

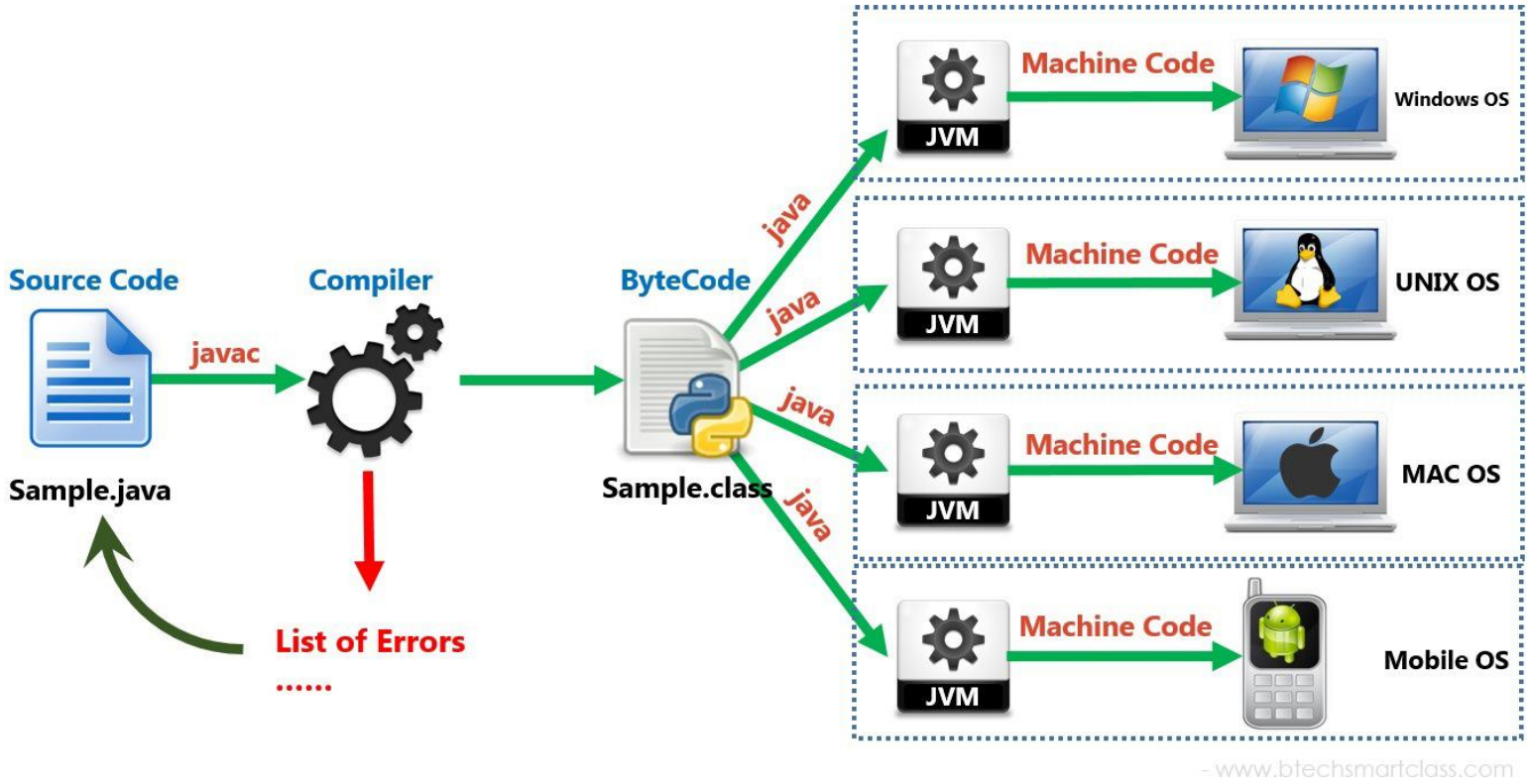
- Java can run on **low-power devices** and **embedded systems**.
  - Smart TVs, Blu-ray players, and IoT (Internet of Things) devices often use Java.
- Java plays a significant role in the big data ecosystem.
  - Tools and frameworks like Apache Hadoop and Apache Spark are built using Java, providing robust solutions for big data processing and analysis.
- Java is frequently used in cloud computing due to its scalability and security features.
  - Many cloud-based applications and platforms rely on Java for data storage, sharing, and complex computations.
- Java can also be used for building desktop applications with graphical user interfaces (GUIs).
  - Applications like media players, antivirus software, and IDEs (like Eclipse) are built using Java.

Key Features of Java

- Java is a simple language
  - It has provided the garbage collector for memory management. It helps in automatically delete unused objects.
- Java is a multi-threaded language:
  - Java can perform many tasks at once by defining multiple threads.
- Java is an object-oriented programming(OOP) language.
- Java is a platform-independent language
  - Java is highly portable because its bytecodes can run on any machine via the Java Virtual Machine (JVM), ensuring code reusability.

Java Code Execution

- **Source Code** (.java file): You write Java code in .java files.
- **Compilation** (javac): The Java compiler (javac) converts the source code into bytecode, producing a .class file.
- **Bytecode** (.class file): This bytecode is platform-independent and can run on any machine with a JVM.
- **Java Virtual Machine** (JVM): The JVM interprets or compiles the bytecode into machine code specific to the platform it runs on.
- **Execution**: The JVM executes the machine code, allowing the program to run.

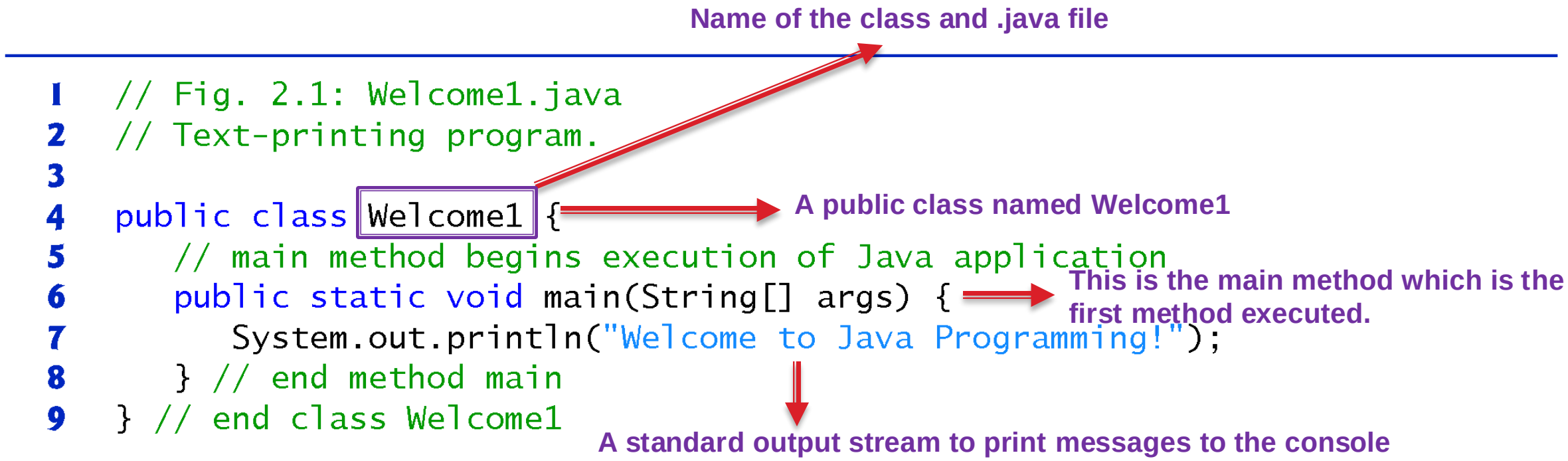


Reference for Image : http://www.btechsmartclass.com/java/java-overview.html

1.9 A Typical Java Development Environment (Cont.)

- **Integrated development environments (IDEs)** provide tools that support the software development process, such as editors, debuggers for locating logic errors that cause programs to execute incorrectly and more.
- The most popular Java IDEs are:
  - Eclipse (http://www.eclipse.org)
  - IntelliJ IDEA (http://www.jetbrains.com)
  - NetBeans (http://www.netbeans.org)

2.2 Your First Program in Java: Printing a Line of Text




```
Welcome to Java Programming!
```

Fig. 2.1 | Text-printing program.

2.2 Your First Program in Java: Printing a Line of Text (Cont.)

Good Programming Practice 2.1

Some organizations require that every program begin with a comment that states the purpose of the program and the author, date and time when the program was last modified.

- Comments

```
// Fig. 2.1: Welcome1.java
```

  - `//` indicates that the line is a comment.
  - Used to document programs and improve their readability.
  - Compiler ignores comments.
  - A comment that begins with `//` is an end-of-line comment—it terminates at the end of the line on which it appears.
- Traditional comment, can be spread over several lines as in

```
/* This is a traditional comment. It
   can be split over multiple lines */
```

  - This type of comment begins with `/*` and ends with `*/`.
  - All text between the delimiters is ignored by the compiler.

Common Programming Error 2.1

Forgetting one of the delimiters of a traditional or Javadoc comment is a syntax error. A **syntax error** occurs when the compiler encounters code that violates Java's language rules (i.e., its syntax). These rules are similar to natural-language grammar rules specifying sentence structure, such as those in English, French, Spanish, etc. Syntax errors are also called **compiler errors**, **compile-time errors** or **compilation errors**, because the compiler detects them when compiling the program. When a syntax error is encountered, the compiler issues an error message. You must eliminate all compilation errors before your program will compile properly.

2.2 Your First Program in Java: Printing a Line of Text (Cont.)

Using Blank Lines

- Blank lines, space characters and tabs
  - Make programs easier to read.
  - Together, they're known as white space (or whitespace).
  - White space is ignored by the compiler.

2.2 Your First Program in Java: Printing a Line of Text (Cont.)

Declaring a class

- Class declaration

  `public class Welcome1`

  - Every Java program consists of at least one class that you define.
  - `class` keyword introduces a class declaration and is immediately followed by the class name.
  - Keywords (Appendix C) are reserved for use by Java and are always spelled with all lowercase letters.

Java Keywords				
abstract	assert	boolean	break	byte
case	catch	char	class	continue
default	do	double	else	enum
extends	final	finally	float	for
if	implements	import	instanceof	int
interface	long	native	new	package
private	protected	public	return	short
static	strictfp	super	switch	synchronized
this	throw	throws	transient	try
void	volatile	while		
Keywords that are not currently used				
const	goto			

Fig. C.1 | Java keywords.

2.2 Your First Program in Java: Printing a Line of Text (Cont.)

Class Names and Identifiers

- By convention, begin with a capital letter and capitalize the first letter of each word they include (e.g., `SampleClassName`).
- A class name is a type of identifier that can include letters, digits, underscores (_), and dollar signs ($), but it cannot start with a digit or contain spaces. Java is case sensitive.
- Uppercase and lowercase letters are distinct—so `a1` and `A1` are different (but both valid) identifiers.

Class Body

- A left brace, `{`, begins the body of every class declaration.
- A corresponding right brace, `}`, must end each class declaration.

Good Programming Practice 2.4

Indent the entire body of each class declaration one "level" between the braces that delimit the class's. This format emphasizes the class declaration's structure and makes it easier to read. We use three spaces to form a level of indent—many programmers prefer two or four spaces. Whatever you choose, use it consistently.

Good Programming Practice 2.5

IDEs typically indent code for you. The Tab key may also be used to indent code. You can configure each IDE to specify the number of spaces inserted when you press Tab.

2.2 Your First Program in Java: Printing a Line of Text (Cont.)

Declaring a Method

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

- Starting point of every Java application.
- Parentheses after the identifier `main` indicate that it's a program building block called a method.
- Java class declarations normally contain one or more methods.
- `main` must be defined as shown; otherwise, the JVM will not execute the application.
- Methods perform tasks and can return information when they complete their tasks.
- Keyword `void` indicates that this method will not return any information.

```
1  // Fig. 2.1: Welcome1.java
2  // Text-printing program.
3
4  public class Welcome1 {
5     // main method begins execution of Java application
6     public static void main(String[] args) {
7        System.out.println("Welcome to Java Programming!");
8     } // end method main
9  } // end class Welcome1
```

2.2 Your First Program in Java: Printing a Line of Text (Cont.)

- Body of the method declaration
  - Enclosed in left and right braces.
- Statement

```
System.out.println("Welcome to Java Programming!");
```

  - Instructs the computer to perform an action
    - Display the characters contained between the double quotation marks.
  - Together, the quotation marks and the characters between them are a string—also known as a character string or a string literal.
  - White-space characters in strings are *not* ignored by the compiler.

2.2 Your First Program in Java: Printing a Line of Text (Cont.)

- `System.out` object
  - Standard output object.
  - Allows a Java application to display information in the command window from which it executes.
- `System.out.println` method
  - Displays (or prints) a line of text in the command window.
  - The string in the parentheses the argument to the method.
  - Positions the output cursor at the beginning of the next line in the command window.
- Most statements end with a semicolon.

2.2 Your First Program in Java: Printing a Line of Text (Cont.)

Compiling Your First Java Application

- Open a command window and change to the directory where the program is stored.
- Many operating systems use the command `cd` to change directories.
- To compile the program, type

```
javac Welcome1.java
```

- If the program contains no compilation errors, preceding command creates a `.class` file (known as the class file) containing the platform-independent Java bytecodes that represent the application.
- When we use the `java` command to execute the application on a given platform, these bytecodes will be translated by the JVM into instructions that are understood by the underlying operating system.

2.2 Your First Program in Java: Printing a Line of Text (Cont.)

Executing the `Welcome1` Application

- To execute this program in a command window, change to the directory containing `Welcome1.java`
  - `C:\examples\ch02\fig02_01` on Microsoft Windows or
  - `~/Documents/examples/ch02/fig02_01` on Linux/macOS.
- Next, type `java Welcome1`.
- This launches the JVM, which loads the `Welcome1.class` file.
- The command *omits* the `.class` file-name extension; otherwise, the JVM will *not* execute the program.
- The JVM calls class `Welcome1`'s `main` method.

Error-Prevention Tip 2.2

When the compiler reports a syntax error, it may not be on the line that the error message indicates. First, check the line for which the error was reported. If you don't find an error on that line, check several preceding lines.

2.3 Modifying Your First Java Program

- Class `Welcome2`, shown in Fig. 2.3, uses two statements to produce the same output as that shown in Fig. 2.1.
- New and key features in each code listing are highlighted.
- `System.out`'s method `print` displays a string.
- Unlike `println`, `print` does not position the output cursor at the beginning of the next line in the command window.
  - The next character the program displays will appear immediately after the last character that `print` displays.

```
1  // Fig. 2.3: Welcome2.java
2  // Printing a line of text with multiple statements.
3
4  public class Welcome2 {
5     // main method begins execution of Java application
6     public static void main(String[] args) {
7        System.out.print("Welcome to ");
8        System.out.println("Java Programming!");
9     } // end method main
10 } // end class Welcome2
```

```
Welcome to Java Programming!
```

Fig. 2.3 | Printing a line of text with multiple statements.

2.3 Modifying Your First Java Program (Cont.)

- Newline characters indicate to `System.out`'s `print` and `println` methods when to position the output cursor at the beginning of the next line in the command window.
- Newline characters are whitespace characters.
- The backslash (`\`) is called an escape character.
  - Indicates a "special character"
- Backslash is combined with the next character to form an escape sequence—`\n` represents the newline character.
- Complete list of escape sequences

  `http://docs.oracle.com/javase/specs/jls/se7/html/jls-3.html#jls-3.10.6.`

Escape sequence	Description
`\n`	Newline. Position the screen cursor at the beginning of the *next* line.
`\t`	Horizontal tab. Move the screen cursor to the next tab stop.
`\r`	Carriage return. Position the screen cursor at the beginning of the *current* line—do *not* advance to the next line. Any characters output after the carriage return *overwrite* the characters previously output on that line.
`\\`	Backslash. Used to print a backslash character.
`\"`	Double quote. Used to print a double-quote character. For example, `System.out.println("\"in quotes\"");` displays `"in quotes"`.

Fig. 2.5 | Some common escape sequences.

```
1  // Fig. 2.4: Welcome3.java
2  // Printing multiple lines of text with a single statement.
3
4  public class Welcome3 {
5     // main method begins execution of Java application
6     public static void main(String[] args) {
7        System.out.println("Welcome\nto\nJava\nProgramming!");
8     } // end method main
9  } // end class Welcome3
```

```
Welcome
to
Java
Programming!
```

Fig. 2.4 | Printing multiple lines of text with a single statement.

2.4 Displaying Text with `printf`

- `System.out.printf` method
  - `f` means "formatted"
  - displays *formatted* data
- Multiple method arguments are placed in a comma-separated list.
- Calling a method is also referred to as invoking a method.
- Java allows large statements to be split over many lines.
  - Cannot split a statement in the middle of an identifier or string.
- Method `printf`'s first argument is a format string
  - May consist of fixed text and format specifiers.
  - Fixed text is output as it would be by `print` or `println`.
  - Each format specifier is a placeholder for a value and specifies the type of data to output.
- Format specifiers begin with a percent sign (%) and are followed by a character that represents the data type.
- Format specifier `%s` is a placeholder for a string.

```
1  // Fig. 2.6: Welcome4.java
2  // Displaying multiple lines with method System.out.printf.
3
4  public class Welcome4 {
5     // main method begins execution of Java application
6     public static void main(String[] args) {
7        System.out.printf("%s%n%s%n", "Welcome to", "Java Programming!");
8     } // end method main
9  } // end class Welcome4
```

```
Welcome to
Java Programming!
```

Fig. 2.6 | Displaying multiple lines with method `System.out.printf`.

2.5 Another Application: Adding Integers

- Integers
  - Whole numbers, like –22, 7, 0 and 1024)
- Programs remember numbers and other data in the computer's memory and access that data through program elements called variables.

```
1  // Fig. 2.7: Addition.java
2  // Addition program that inputs two numbers then displays their sum.
3  import java.util.Scanner; // program uses class Scanner
4
5  public class Addition {
6     // main method begins execution of Java application
7     public static void main(String[] args) {
8        // create a Scanner to obtain input from the command window
9        Scanner input = new Scanner(System.in);
10
11       System.out.print("Enter first integer: "); // prompt
12       int number1 = input.nextInt(); // read first number from user
13
14       System.out.print("Enter second integer: "); // prompt
15       int number2 = input.nextInt(); // read second number from user
16
17       int sum = number1 + number2; // add numbers, then store total in sum
18
19       System.out.printf("Sum is %d%n", sum); // display sum
20    } // end method main
21 } // end class Addition
```

```
Enter first integer: 45
Enter second integer: 72
Sum is 117
```

Fig. 2.7 | Addition program that inputs two numbers then, displays their sum. (Part 1 of 2.)

2.5 Another Application: Adding Integers (cont.)

- import helps the compiler locate a class that is used in this program.
- Rich set of predefined classes that you can reuse rather than "reinventing the wheel."
- You use `import` declarations to identify the predefined classes used in a Java program.

```
1  // Fig. 2.7: Addition.java
2  // Addition program that inputs two numbers then displays their sum.
3  import java.util.Scanner; // program uses class Scanner
```

Common Programming Error 2.5

All `import` declarations must appear before the first class declaration in the file. Placing an `import` declaration inside or after a class declaration is a syntax error.

2.5 Another Application: Adding Integers (cont.)

- `Scanner`
  - Enables a program to read data for use in a program.
  - Data can come from many sources, such as the user at the keyboard or a file on disk.
  - Before using a `Scanner`, you must create it and specify the source of the data.
- The equals sign (=) in a declaration indicates that the variable should be initialized (i.e., prepared for use in the program) with the result of the expression to the right of the equals sign.
- The `new` keyword creates an object.
- Standard input object, `System.in`, enables applications to read bytes of data typed by the user.
- `Scanner` object translates these bytes into types that can be used in a program.
- Class System
  - Unlike other classes, System does not require an import declaration at the beginning of the program.
  - The java.lang package is imported by default, meaning classes like System, String, and Math can be used directly without an explicit import statement.
  - This class provides methods for input and output, including System.out.print() and System.out.println() for displaying text.

2.5 Another Application: Adding Integers (cont.)

- Variable declaration statement

```
int number1 = input.nextInt(); // read first number from user
```

  declares that variables `number1` holds data of type `int`
  - Range of values for an `int` is –2,147,483,648 to +2,147,483,647.
- `Scanner` method `nextInt`
  - Obtains an integer from the user at the keyboard.
  - Program *waits* for the user to type the number and press the *Enter* key to submit the number to the program.
- The result of the call to method `nextInt` is placed in variable `number1`
  - The = indicates that `int` variable `number1` should be initialized in its declaration with the result of `input.nextInt()`
- Integer formatted output

```
System.out.printf("Sum is %d%n", sum);
```

  - Format specifier `%d` is a *placeholder* for an `int` value
  - The letter d stands for “decimal integer.”

Questions?