Java Introduction

Teacher:
Nasrin Khodapanah

Good to know

Java is a programming language and computing platform that is widely used to develop a wide range of applications, including web, mobile, desktop, and enterprise applications. It is known for its "write once, run anywhere" philosophy, which means that Java code can be compiled and run on any platform that has a Java Virtual Machine (JVM) installed.

The Java structure consists of several key components:

The Java Virtual Machine (JVM): This is the core component of the Java platform that provides the runtime environment for executing Java code. The JVM interprets compiled Java code and manages memory and other resources.
The Java Development Kit (JDK): This is a set of tools and libraries that developers use to create, test, and debug Java applications. The JDK includes the JVM, a compiler, and other tools such as the Java Debug Wire Protocol (JDWP) and the Java Platform Debugger Architecture (JPDA).
The Java Standard Edition (Java SE): This is the core set of libraries and APIs that form the basis of the Java platform. It includes classes for basic data types, collections, concurrency, networking, and more.

Java Libraries/Packages

In Java, a library is a collection of pre-written and tested classes and interfaces that can be used to perform a variety of tasks. A package is a way of grouping related classes and interfaces and providing a namespace for them.

There are some key differences between Java libraries and packages:

A library is a collection of classes and interfaces that can be used to perform specific tasks, while a package is a grouping mechanism for organizing related classes and interfaces.

A library can contain multiple packages, but a package cannot contain multiple libraries.

Libraries are usually developed and maintained by different organizations or individuals and can be added to a project as needed, while packages are usually developed and maintained by the Java community as part of the Java platform.

Libraries are typically added to a project by including them in the classpath or by importing them into the code, while packages are added to a project by including them in the code using the package keyword.

Libraries are usually external to the project, packages are part of the project.

In summary: libraries are collections of pre-written, tested, and reusable code that can be used to perform specific tasks, while packages are a way of grouping related classes and interfaces and providing a namespace for them.

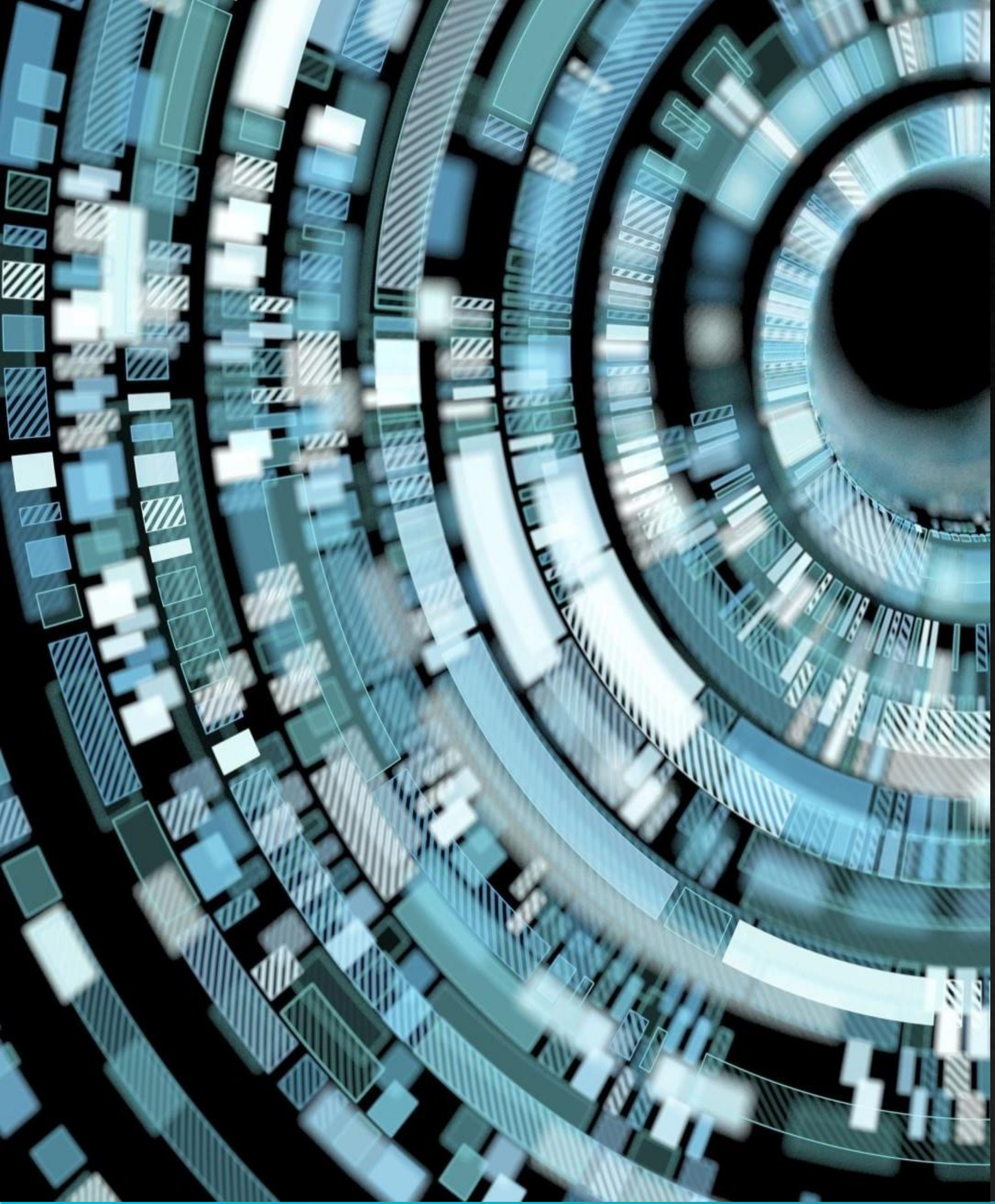
Common Java Libraries and Packages

libraries

- Swing: a Java library used for creating graphical user interfaces (GUIs)
- Servlet: a Java library used for creating web applications and handling HTTP requests and responses
- JPA (Java Persistence API): a Java library for managing relational data in applications
- JDBC (Java Database Connectivity): a Java library for connecting to and interacting with databases
- JUnit: a unit testing framework for Java.
- Spring Framework: A Java framework for building applications using inversion of control and aspect-oriented programming.
- Apache Struts: A Java framework for building web applications using the Model-View-Controller (MVC) design pattern.

packages

- java.util: contains various utility classes for working with data, such as the Collections class for working with collections, and the Date class for working with dates and times.
- java.io: contains classes for input and output operations, such as the File class for working with files and the PrintWriter class for writing to a file.
- java.lang: contains fundamental classes and interfaces that form the core of the Java language, such as the String class for working with strings, and the Object class, which is the root of the class hierarchy.
- java.net: contains classes for working with network operations, such as the URL class for working with URLs and the Socket class for working with sockets.
- Java.awt: contains the classes and interfaces for creating a platform-independent windowing, graphics, and user-interface widgets. It stands for "Abstract Window Toolkit", and is the original Java GUI package, it provides a basic set of tools for creating graphical user interfaces. The Abstract Window Toolkit (AWT) provides the foundation for creating Graphical User Interfaces (GUIs) in Java.
- Javax.swing: is an extension of the java.awt package that provides a more sophisticated set of GUI components. It stands for "Swing Extension Package". It is an extension of the AWT that provides a richer set of graphical user interface (GUI) components than the AWT.



Object-oriented programming (OOP)

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which can contain data and code that manipulates that data. In Java, OOP is implemented using classes, which are templates for creating objects. Objects are instances of a class and can have their state and behavior. Classes have properties (also called fields or attributes) and methods, which are functions that operate on the object's data(We talk about it in Module III)

..some top-level classes

Module II

- Object class:

  Properties: none

  Methods: equals(), toString(), hashCode(), getClass()

  - String class (subclass of Object):

  Properties: length()

  Methods: toUpperCase(), substring(), concat(), trim()

  - Number class (subclass of Object):

  Properties: valueOf()

  Methods: intValue(), longValue(), floatValue(), doubleValue()

    - Integer class (subclass of Number):
      Properties: MIN_VALUE, MAX_VALUE
      Methods: parseInt(), toString(), valueOf(), compareTo()

- ArrayList class (subclass of AbstractList and List interfaces)

Properties: size()

Methods: add(), get(), remove(), clear(), contains()

JFrame and JPanel are both classes in the Java Swing Package, which is used for creating graphical user interfaces (GUIs) in Java.

JFrame class:
Properties: setSize(), setTitle(), setDefaultCloseOperation()
Methods: add(), setVisible(), pack(), dispose()

JPanel class:
Properties: none
Methods: add(), setBackground(), setLayout(), setPreferredSize()

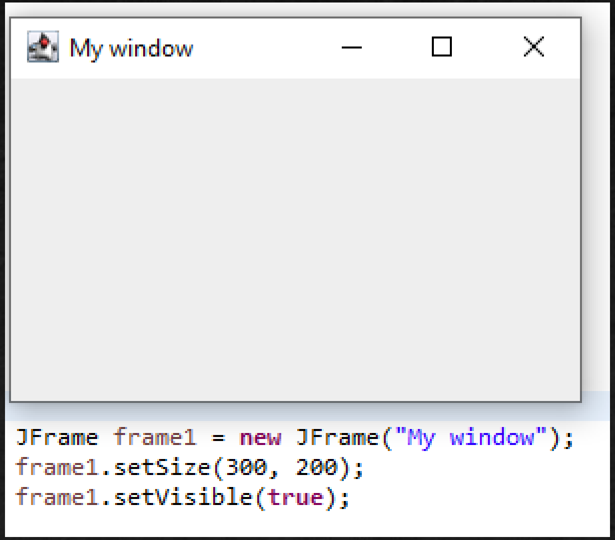
JFrame is used to create the main window of a GUI application, and it can contain other components such as JPanel, JButton, JLabel, etc. JPanel is used as a container to hold other components, and it can be added to a JFrame or another container.

Using Classes in Java:

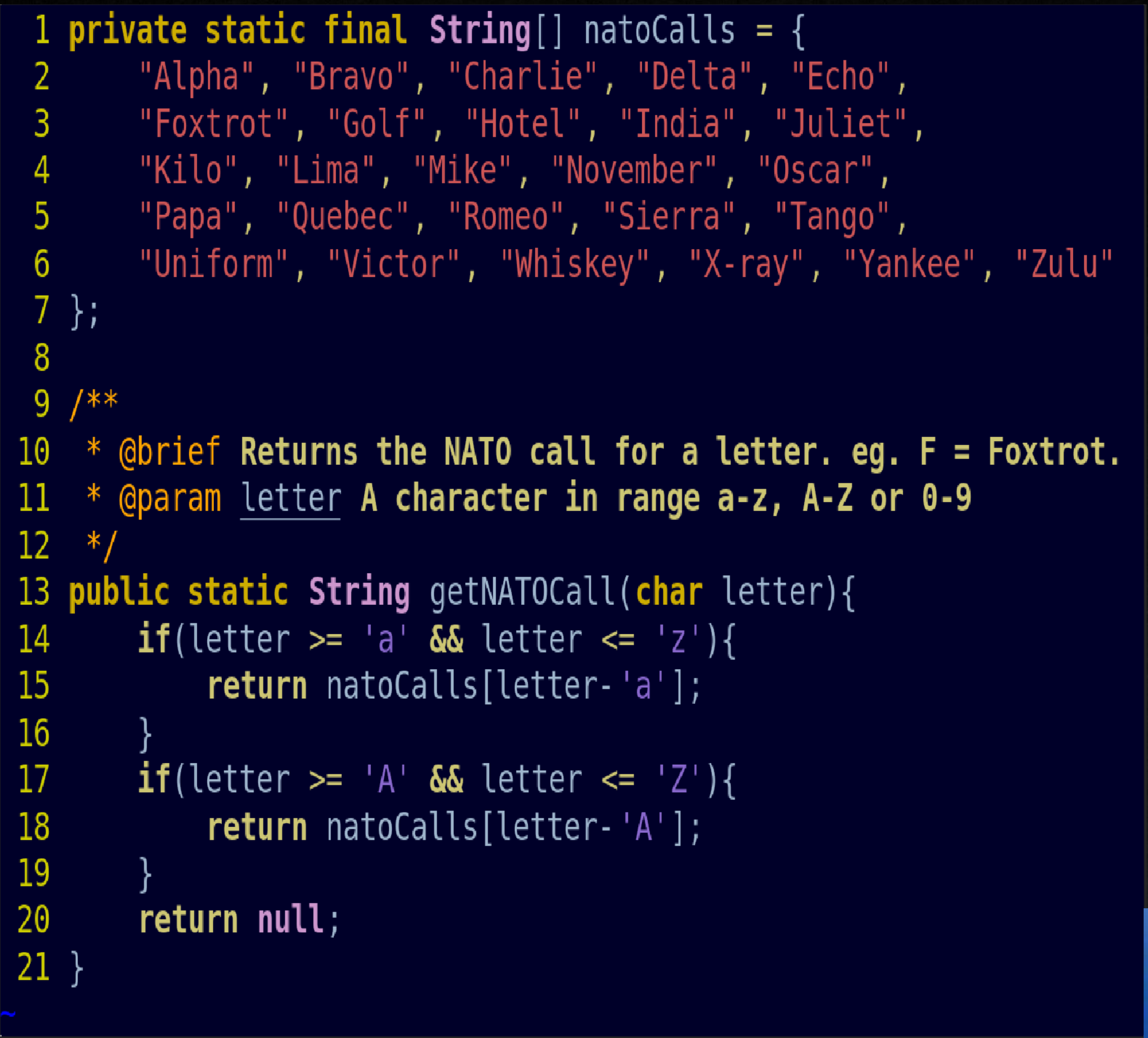
Importing, Instantiating, and Utilizing Classes in your Application

```
import javax.swing.*;

public class MyWindow {
    public static void main(String[]
args) {
JFrame frame = new JFrame("My Window");
frame.setSize(300, 200);
frame.setVisible(true);

    }
}
```

- To use a class in a Java application, you first need to import the class using the import keyword. For example `import javax.swing.JFrame;`
- Next, you need to create an instance of the class using the new keyword.
- Once you have created an instance of a class, you can use its methods and fields to perform various actions or store data.



```
JFrame frame1 = new JFrame("My window");
frame1.setSize(300, 200);
frame1.setVisible(true);
```



```java
private static final String[] natoCalls = {
    "Alpha", "Bravo", "Charlie", "Delta", "Echo",
    "Foxtrot", "Golf", "Hotel", "India", "Juliet",
    "Kilo", "Lima", "Mike", "November", "Oscar",
    "Papa", "Quebec", "Romeo", "Sierra", "Tango",
    "Uniform", "Victor", "Whiskey", "X-ray", "Yankee", "Zulu"
};

/**
 * @brief Returns the NATO call for a letter. eg. F = Foxtrot.
 * @param letter A character in range a-z, A-Z or 0-9
 */
public static String getNATOCall(char letter){
    if(letter >= 'a' && letter <= 'z'){
        return natoCalls[letter-'a'];
    }
    if(letter >= 'A' && letter <= 'Z'){
        return natoCalls[letter-'A'];
    }
    return null;
}
~
```

Conclusion

In the first two modules of this block, the focus will be on the basics of Java and Swing.

Primary information about classes, libraries, and packages has been provided in this presentation.

Popular and useful objects like Array List, Random, Scanner, and JFrame will be utilized in Modules 1 and 2 to acquaint you with syntax and enhance your understanding of upcoming Java programming courses.

Module 3 will delve deeper into Object-Oriented Programming (OOP) and cover all topics related to classes.