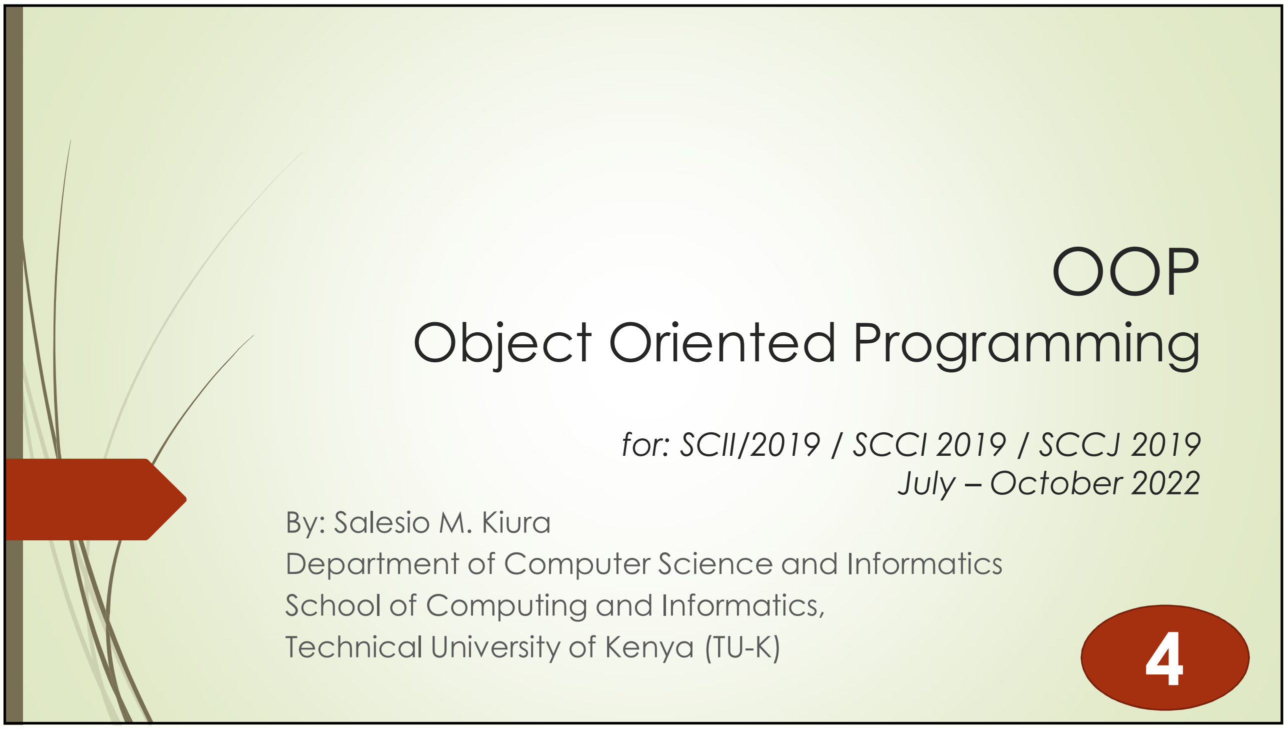
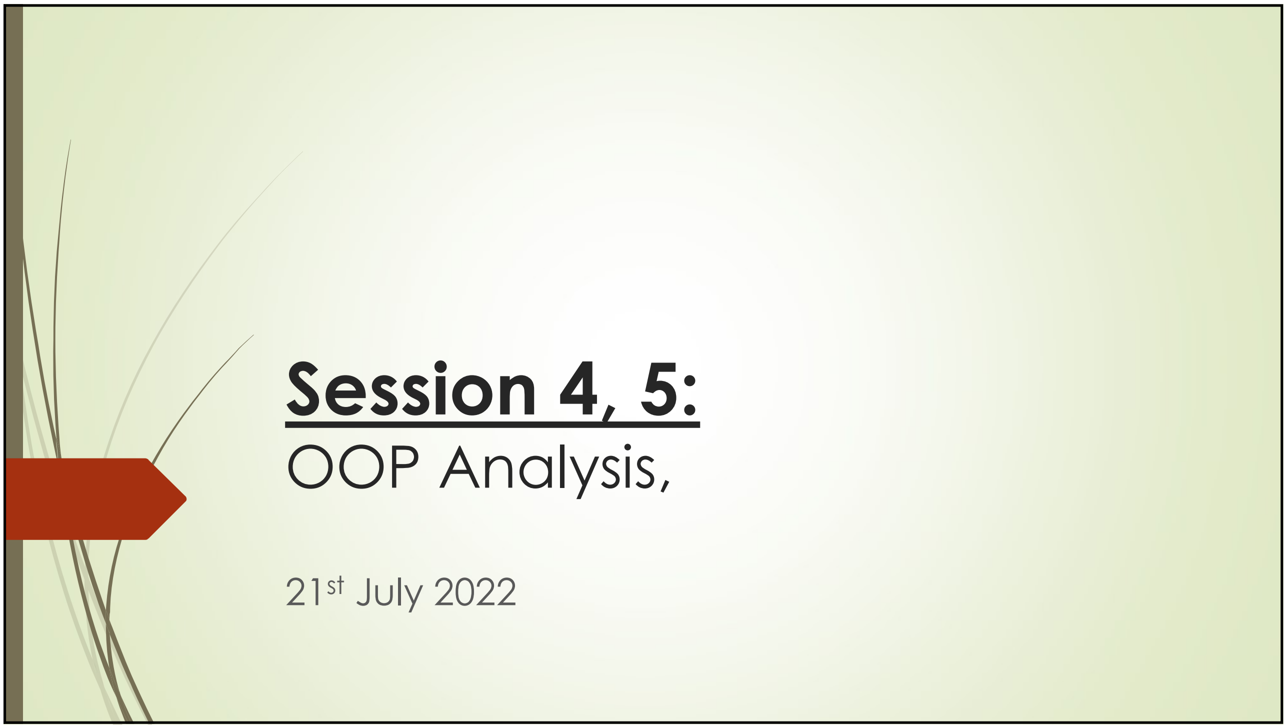


OOP
Object Oriented Programming

for: SCII/2019 / SCCI 2019 / SCCJ 2019
July – October 2022

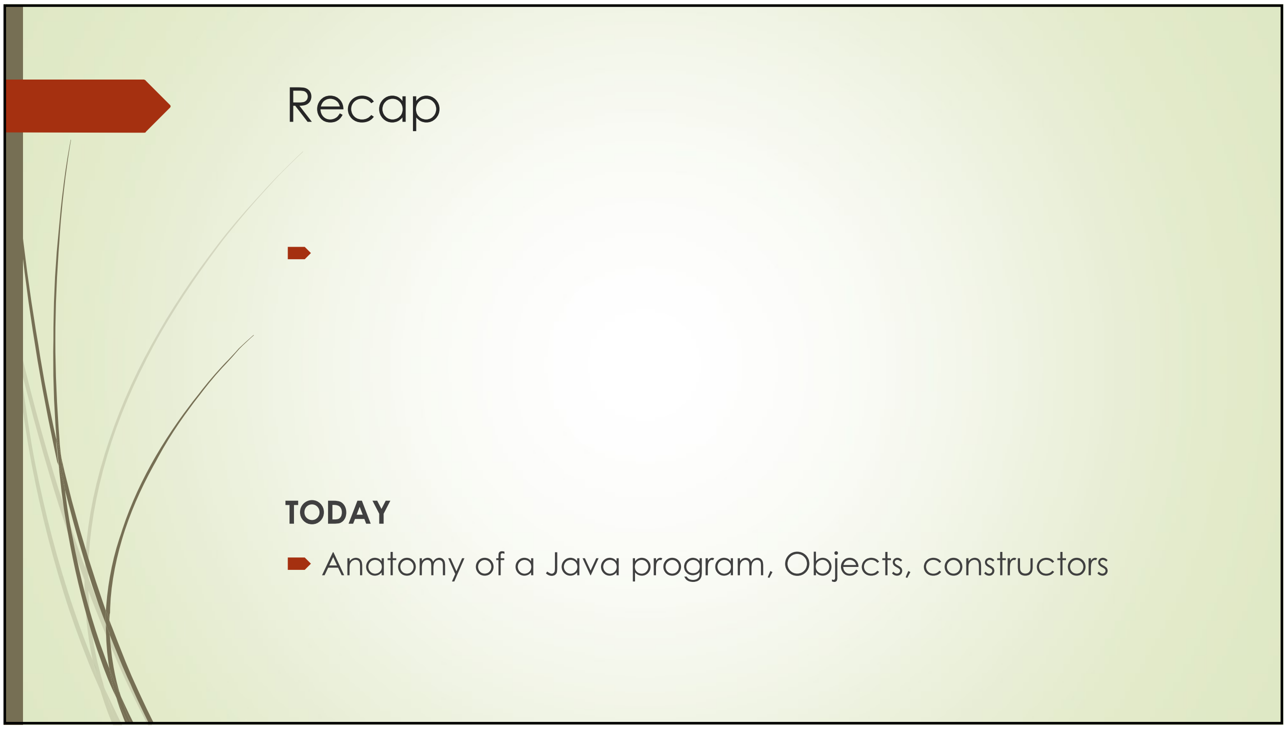
By: Salesio M. Kiura
Department of Computer Science and Informatics
School of Computing and Informatics,
Technical University of Kenya (TU-K)

4



Session 4, 5:
OOP Analysis,

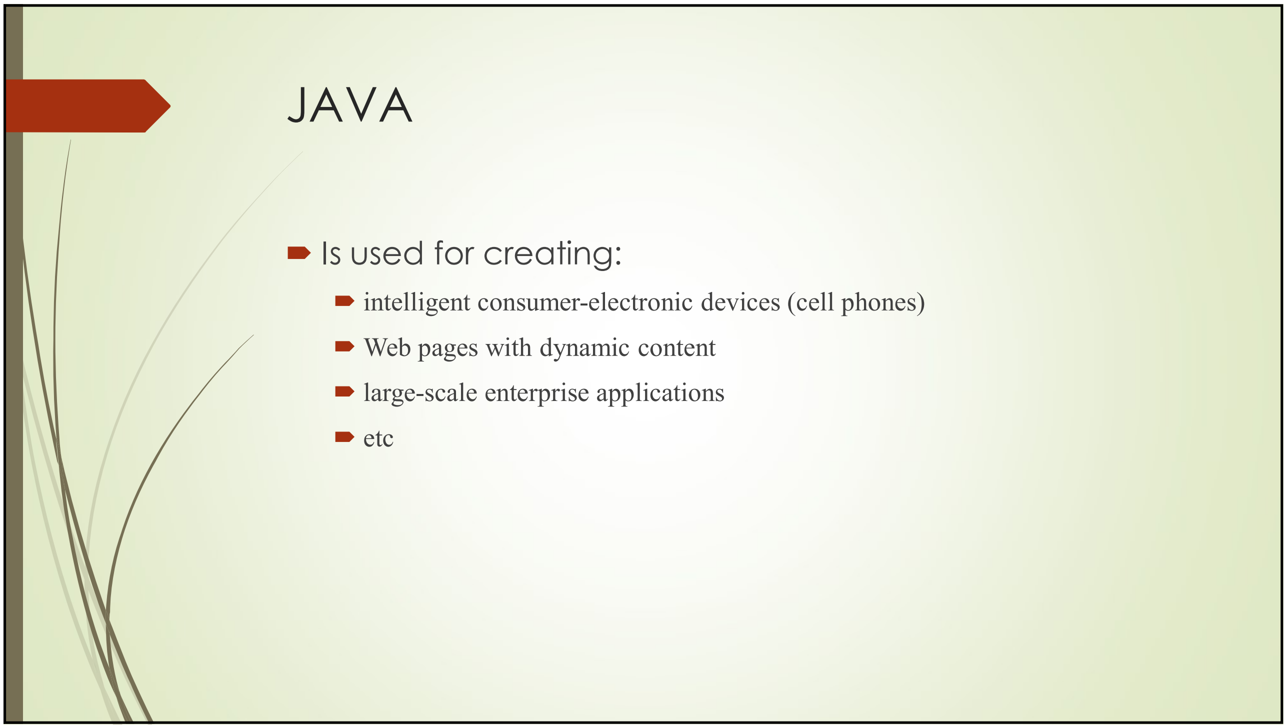
21st July 2022



Recap

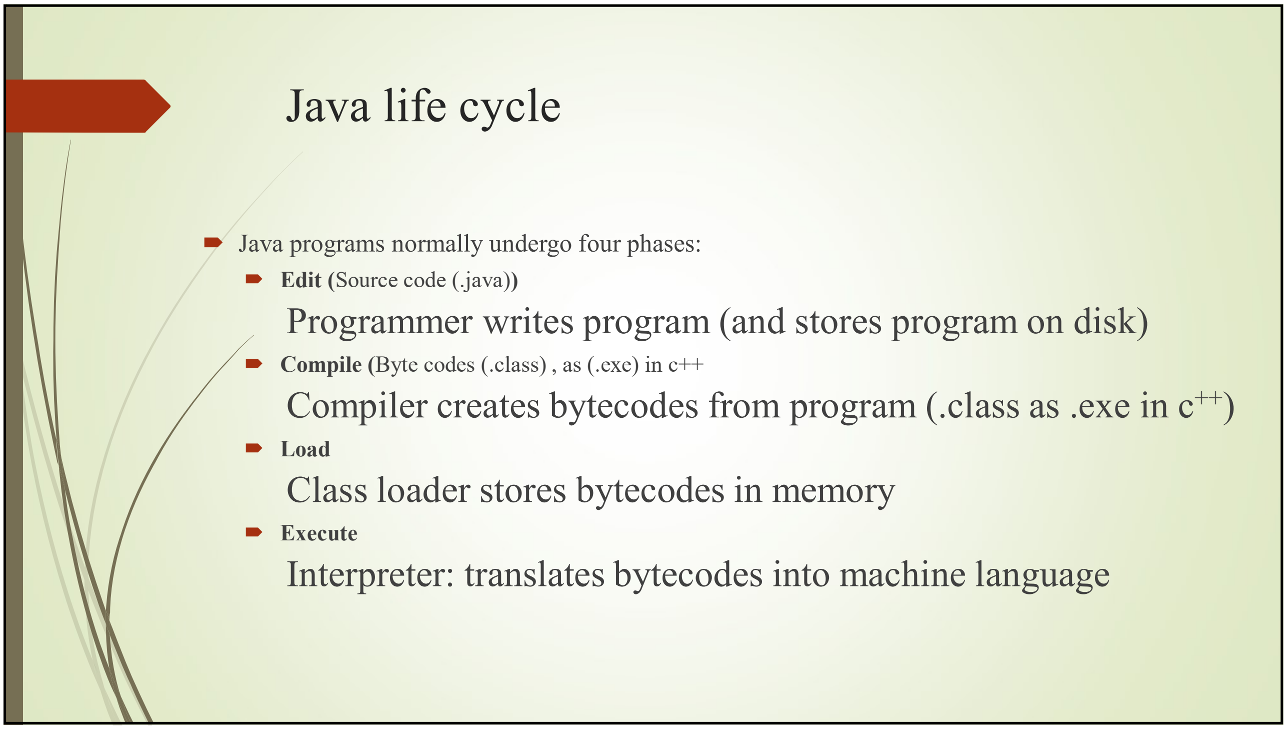
TODAY

- Anatomy of a Java program, Objects, constructors



JAVA

- Is used for creating:
  - intelligent consumer-electronic devices (cell phones)
  - Web pages with dynamic content
  - large-scale enterprise applications
  - etc



Java life cycle

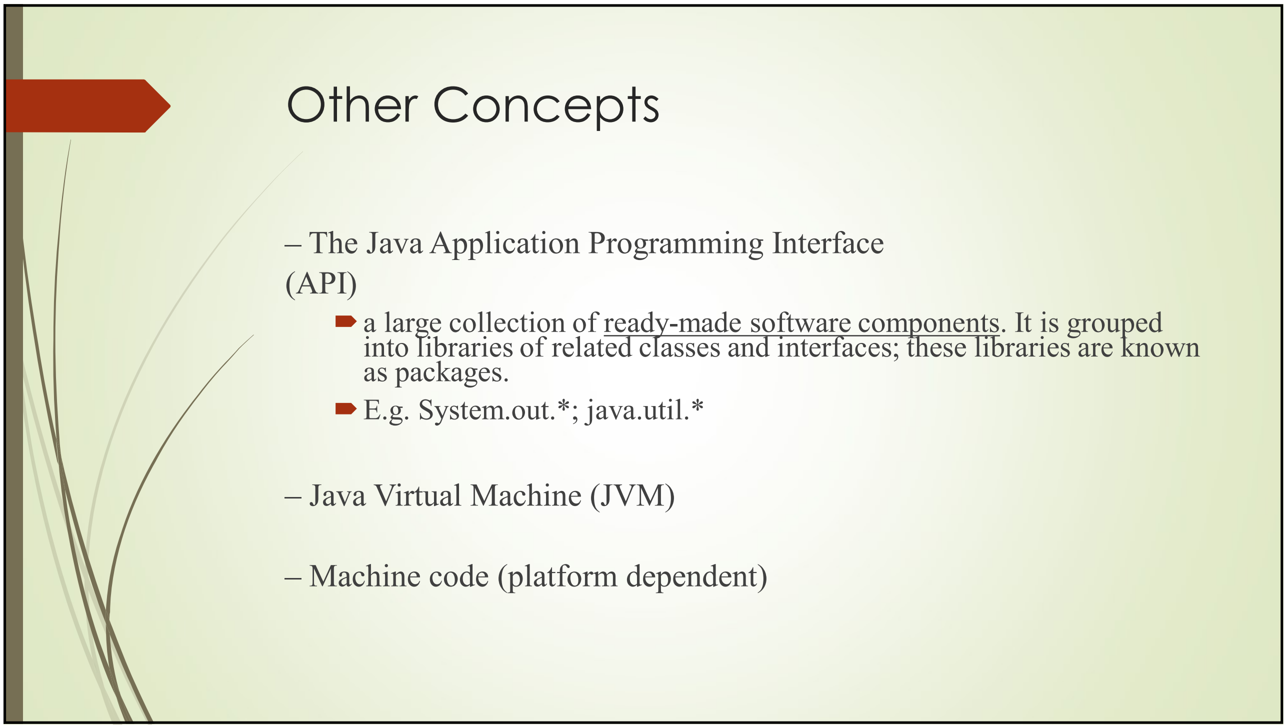
- Java programs normally undergo four phases:
  - **Edit (**Source code (.java)**)**

    Programmer writes program (and stores program on disk)
  - **Compile (**Byte codes (.class) , as (.exe) in c++

    Compiler creates bytecodes from program (.class as .exe in c^{++})
  - **Load**

    Class loader stores bytecodes in memory
  - **Execute**

    Interpreter: translates bytecodes into machine language



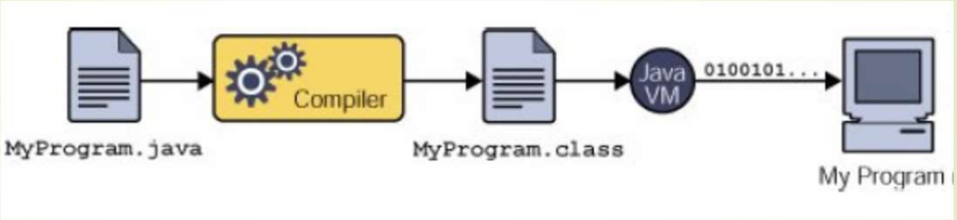
Other Concepts

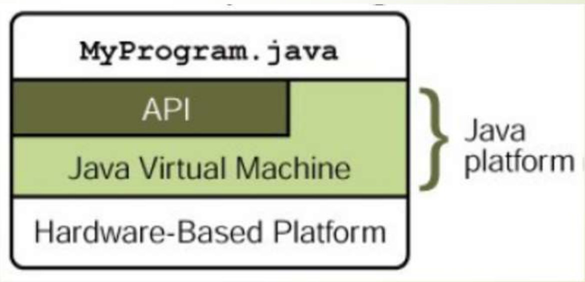
– The Java Application Programming Interface (API)

- a large collection of ready-made software components. It is grouped into libraries of related classes and interfaces; these libraries are known as packages.
- E.g. System.out.*; java.util.*

– Java Virtual Machine (JVM)

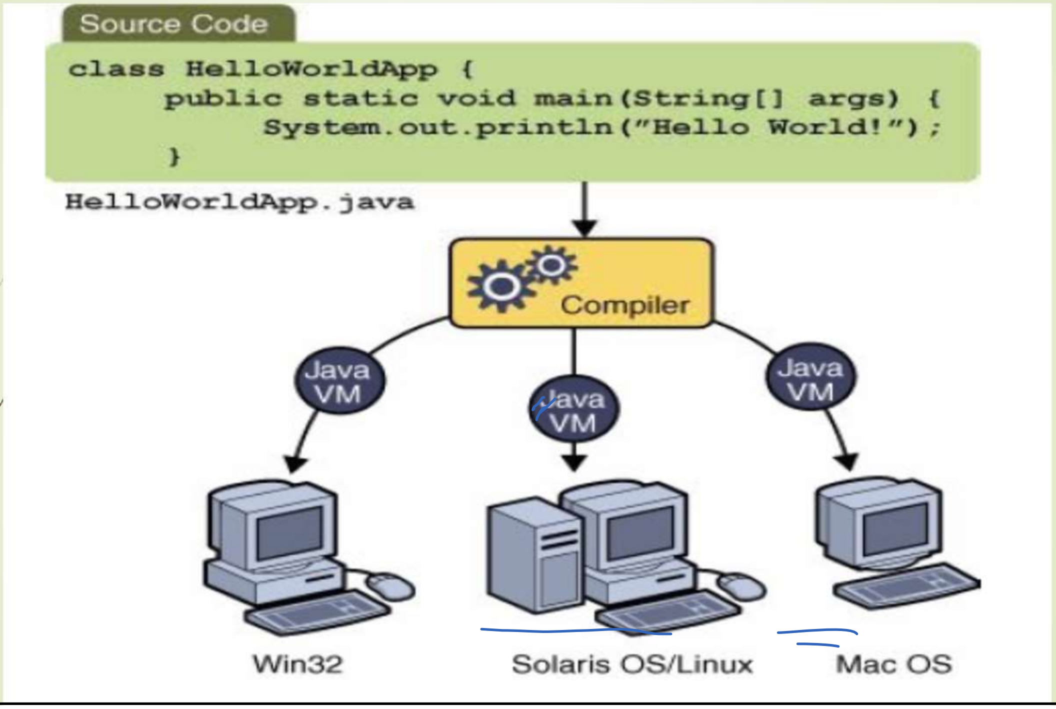
– Machine code (platform dependent)





Portability of Java

- Through the Java VM, the same application is capable of **<u>running on multiple platforms</u>**



```
class HelloWorldApp {
     public static void main(String[] args) {
          System.out.println("Hello World!");
     }
```

Java technology series

- Java EE vs. Java SE

– EE: enterprise edition (web services, distribution, RMI, …)

– SE: standard edition (stand alone applications)

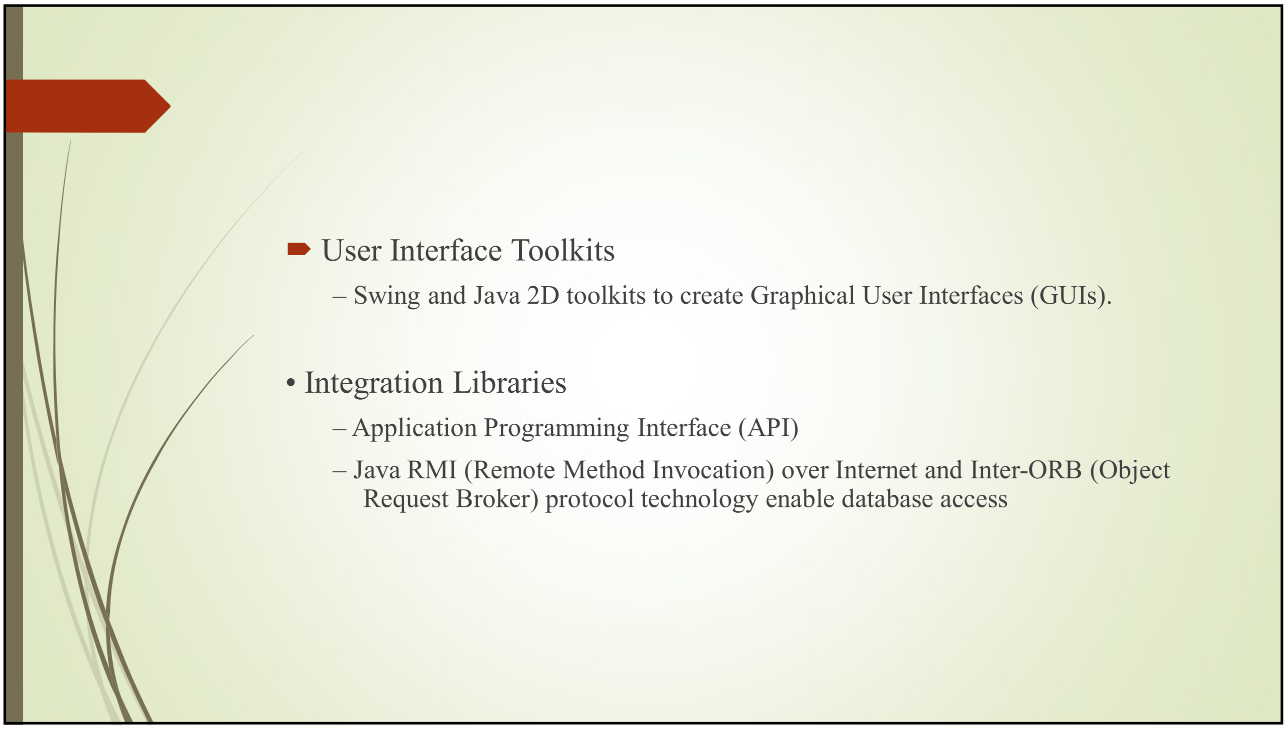
Development Tools

– The main tools you'll be using are the javac compiler, the java launcher (java), and the javadoc documentation tool.

Application Programming Interface (API)

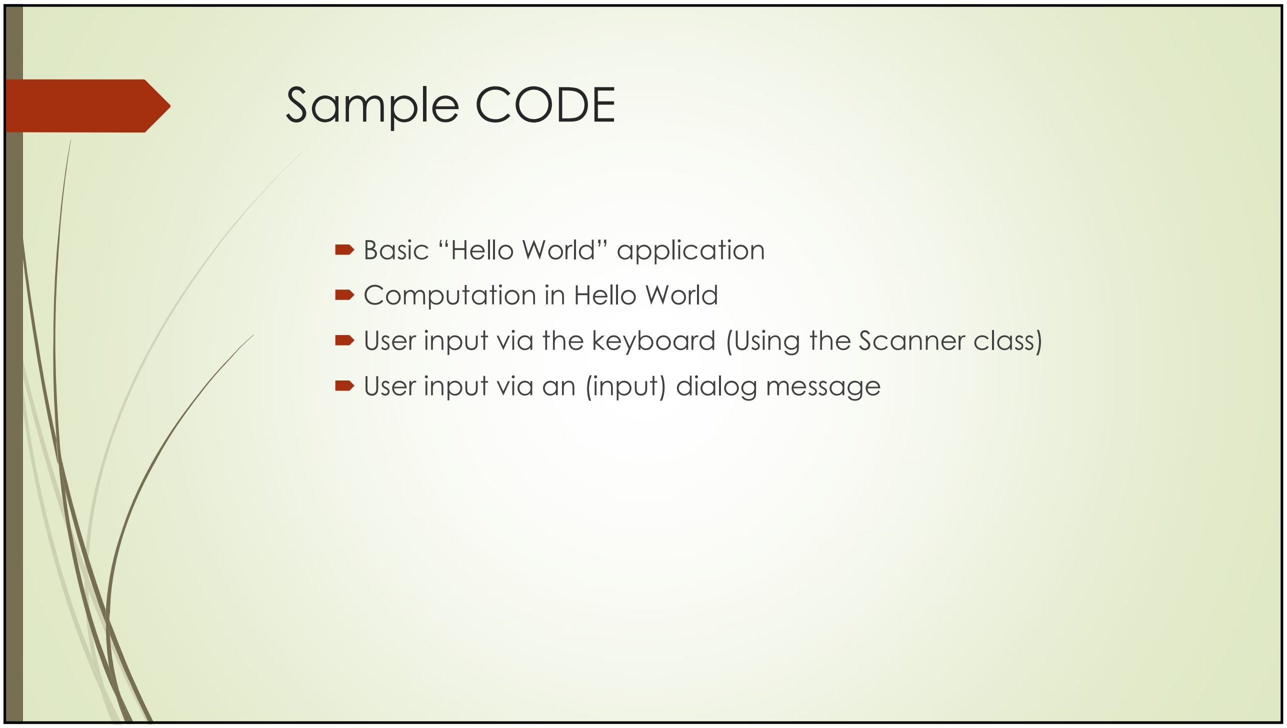
– Java SE Development Kit 8 (JDK 1.8)

– Offers a wide array of useful classes ready for use in your own applications.



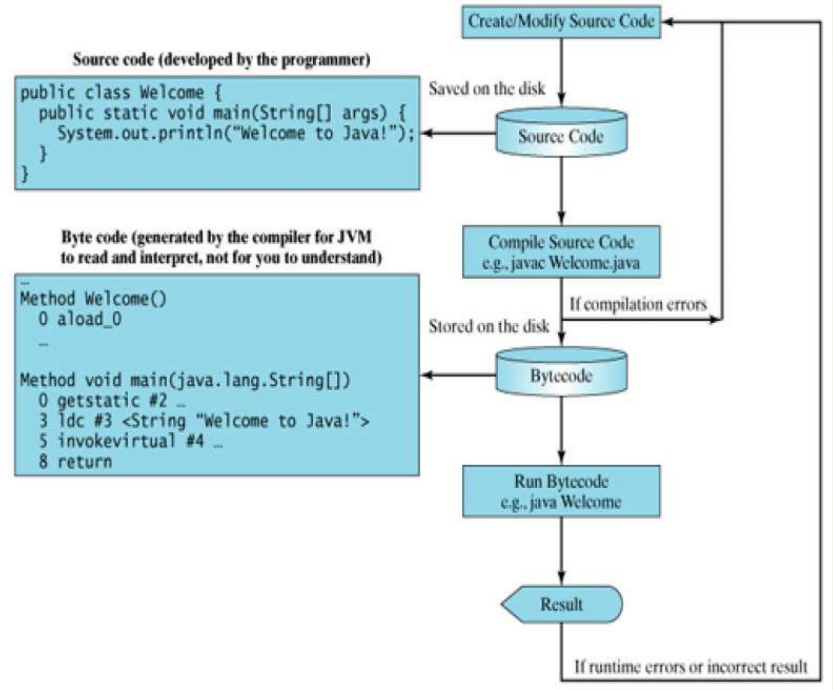
- User Interface Toolkits
  - – Swing and Java 2D toolkits to create Graphical User Interfaces (GUIs).

- Integration Libraries
  - – Application Programming Interface (API)
  - – Java RMI (Remote Method Invocation) over Internet and Inter-ORB (Object Request Broker) protocol technology enable database access



Sample CODE

- Basic “Hello World” application
- Computation in Hello World
- User input via the keyboard (Using the Scanner class)
- User input via an (input) dialog message



Source code (developed by the programmer)

```
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Byte code (generated by the compiler for JVM to read and interpret, not for you to understand)

```
…
Method Welcome()
  0 aload_0
  …

Method void main(java.lang.String[])
  0 getstatic #2 …
  3 ldc #3 <String "Welcome to Java!">
  5 invokevirtual #4 …
  8 return
```

Create/Modify Source Code

Saved on the disk

Source Code

Compile Source Code
e.g., javac Welcome.java

If compilation errors

Stored on the disk

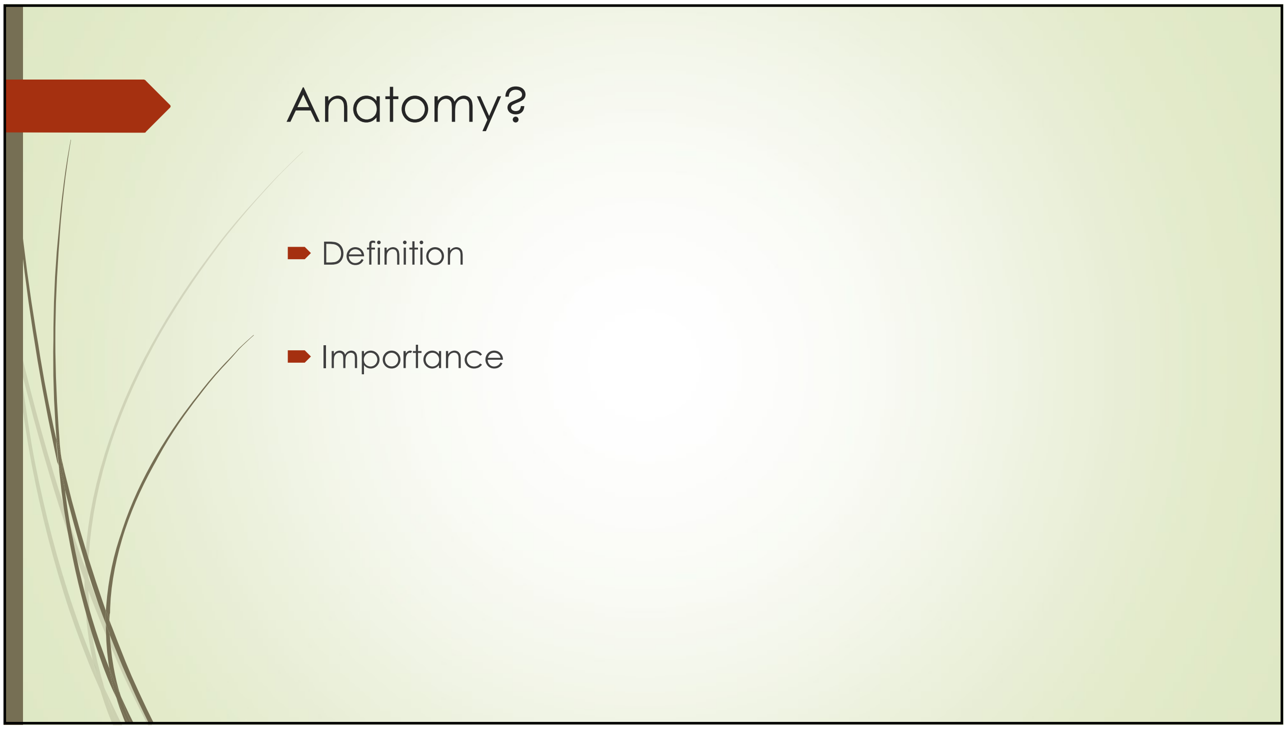
Bytecode

Run Bytecode
e.g., java Welcome

Result

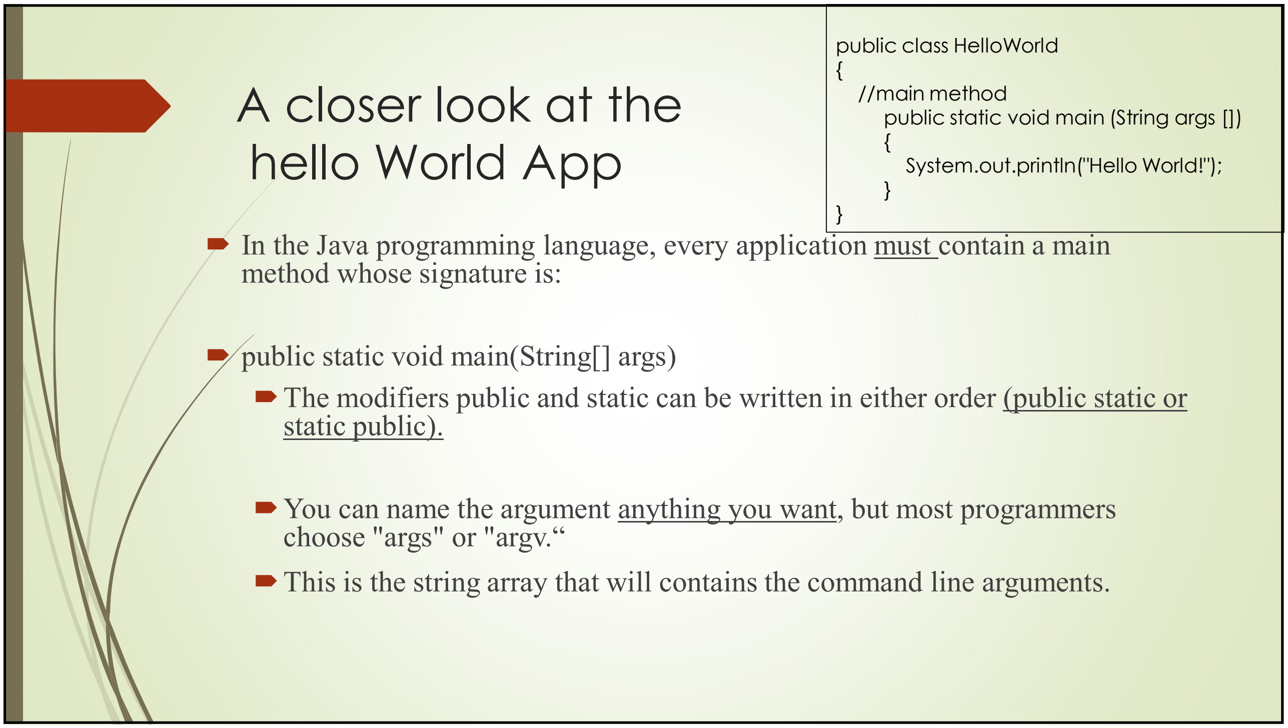
If runtime errors or incorrect result

Anatomy of a Java Application



Anatomy?

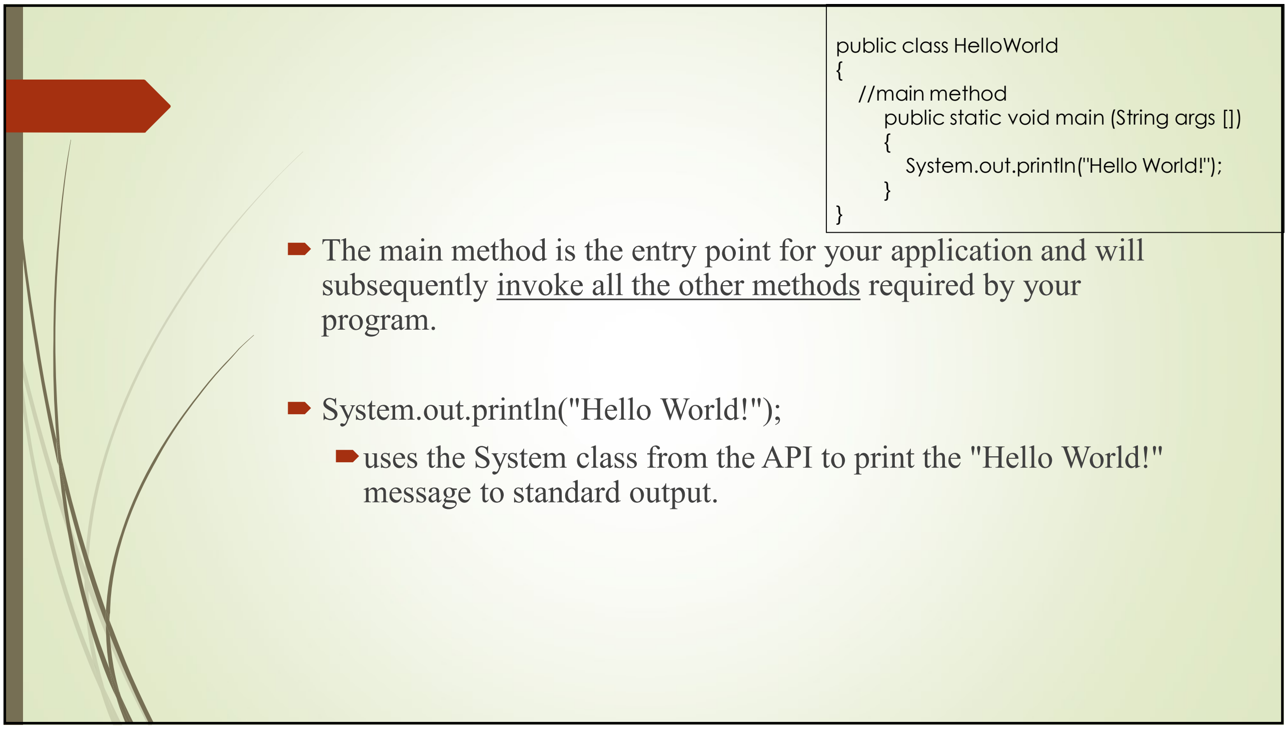
- Definition
- Importance



A closer look at the hello World App

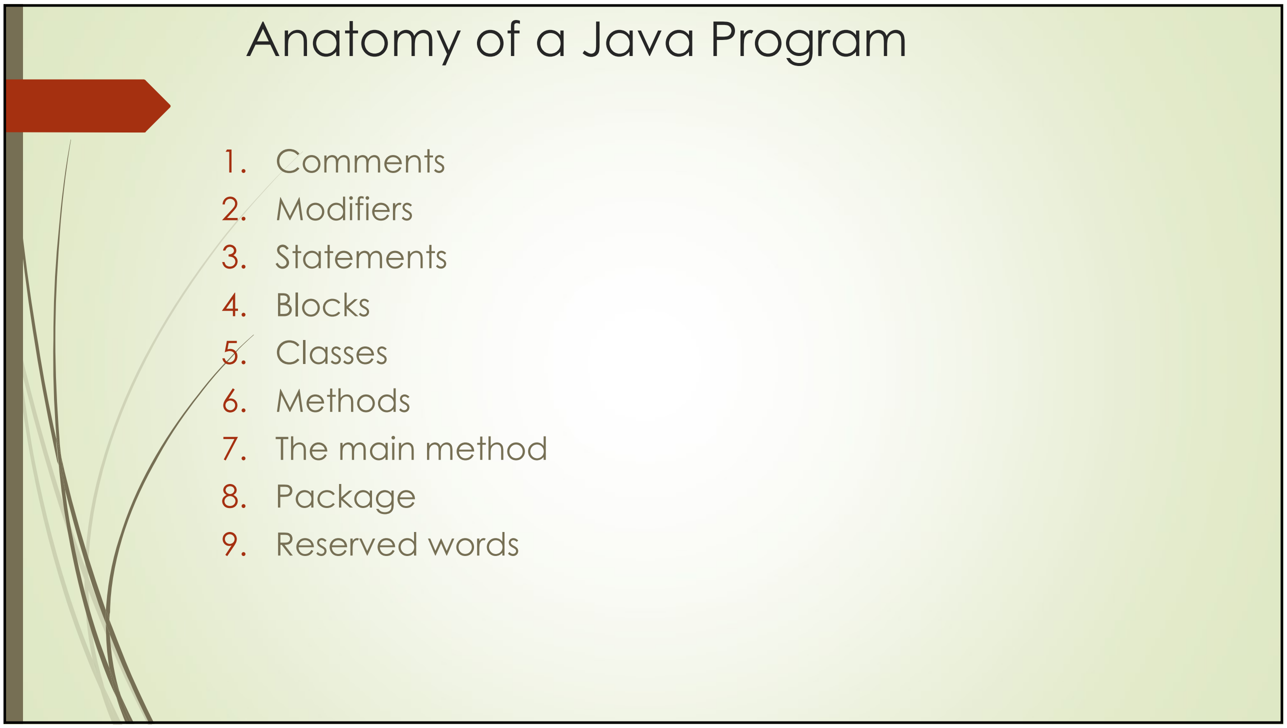
```
public class HelloWorld
{
   //main method
       public static void main (String args [])
       {
          System.out.println("Hello World!");
       }
}
```

- In the Java programming language, every application must contain a main method whose signature is:
- public static void main(String[] args)
  - The modifiers public and static can be written in either order (public static or static public).
  - You can name the argument anything you want, but most programmers choose "args" or "argv.“
  - This is the string array that will contains the command line arguments.



```
public class HelloWorld
{
   //main method
      public static void main (String args [])
      {
         System.out.println("Hello World!");
      }
}
```

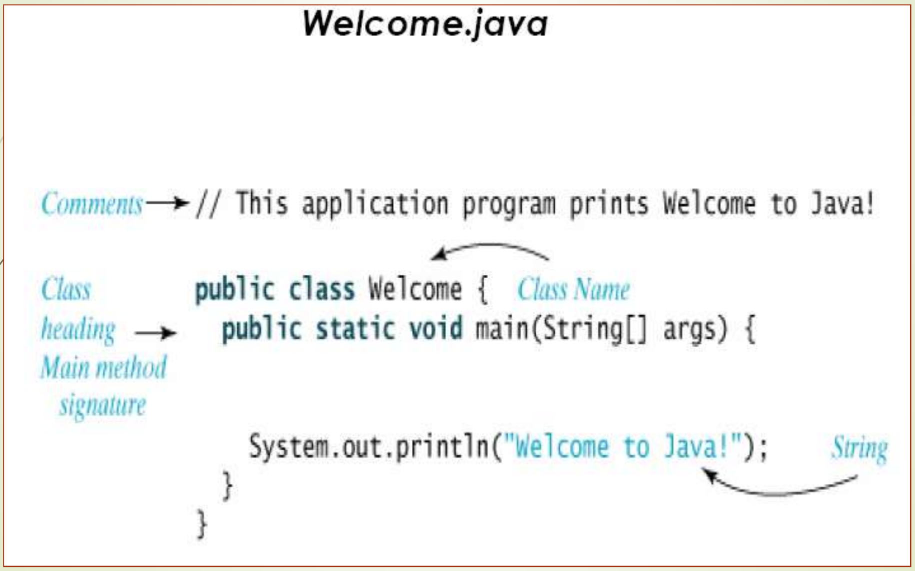
- The main method is the entry point for your application and will subsequently invoke all the other methods required by your program.
- System.out.println("Hello World!");
  - uses the System class from the API to print the "Hello World!" message to standard output.



Anatomy of a Java Program

1. Comments
2. Modifiers
3. Statements
4. Blocks
5. Classes
6. Methods
7. The main method
8. Package
9. Reserved words

Anatomy of a Java Program



Welcome.java

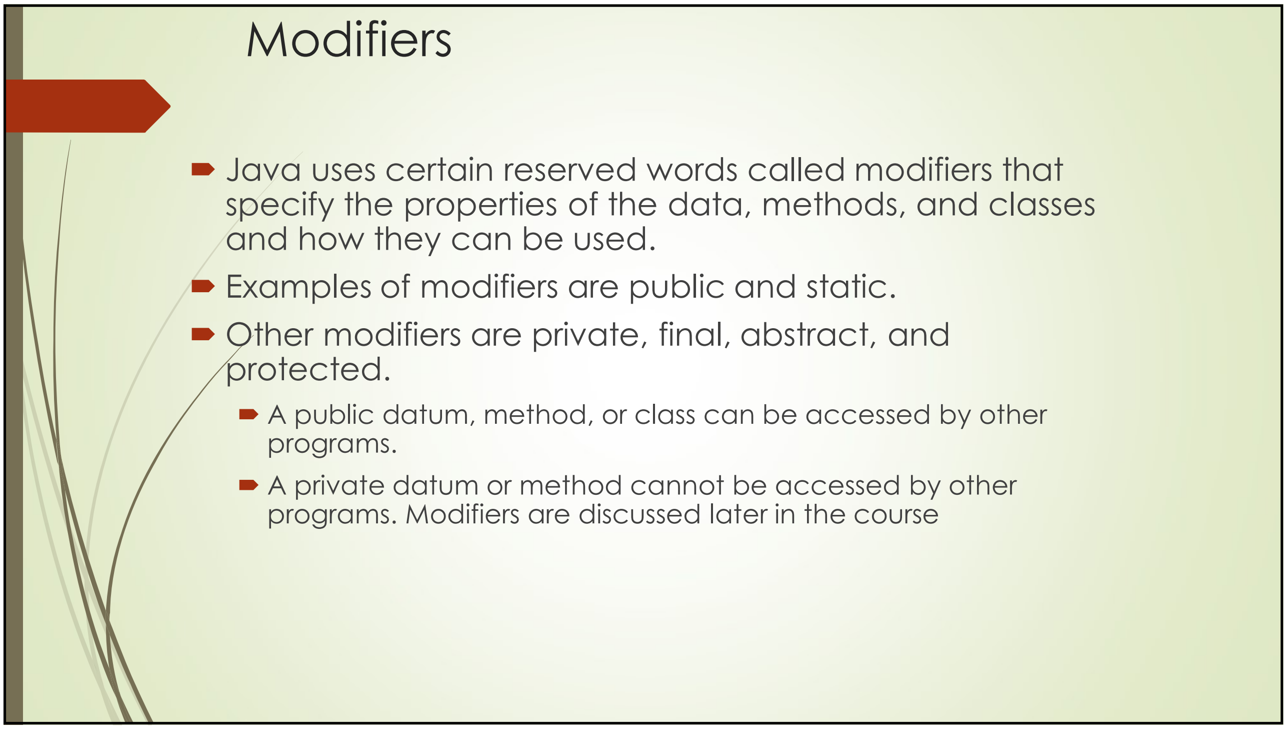
```
// This application program prints Welcome to Java!

public class Welcome {
  public static void main(String[] args) {

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

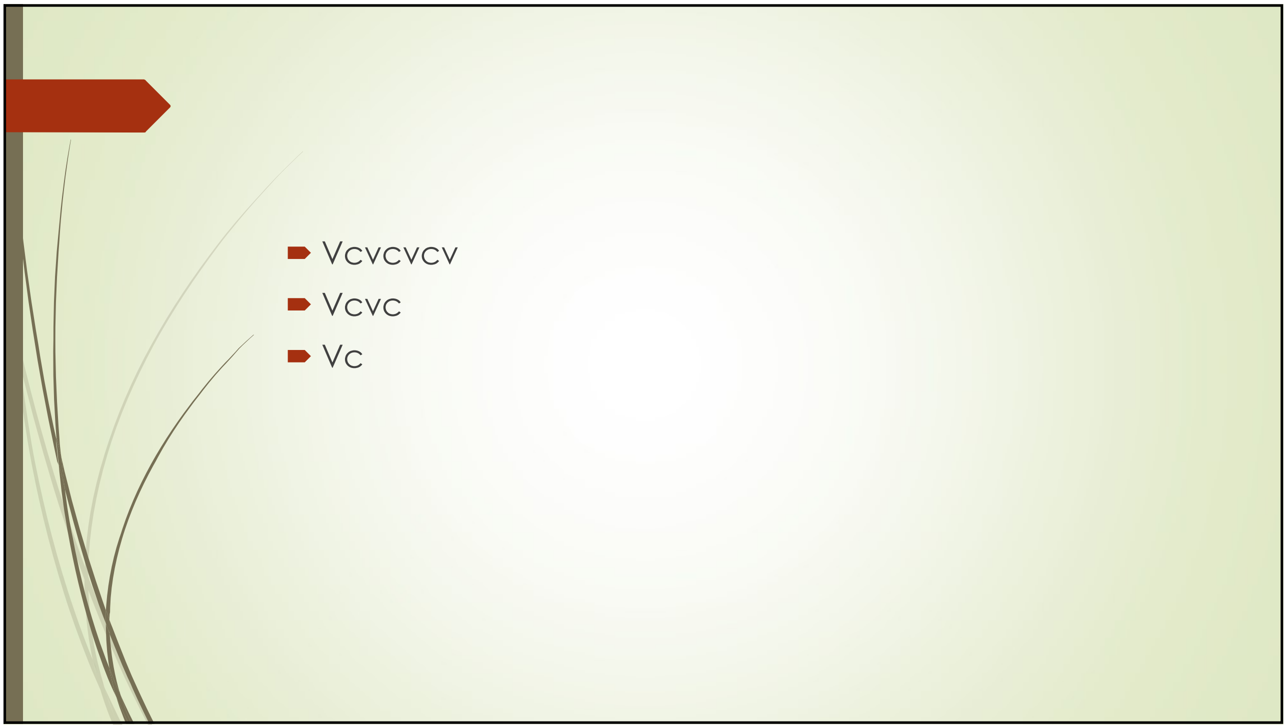
Comments

- In Java, comments are preceded by two slashes (//) in a line, or enclosed between /* and */ in one or multiple lines.
- When the compiler sees //, it ignores all text after // in the same line.
- When it sees /*, it scans for the next */ and ignores any text between /* and */.



Modifiers

- Java uses certain reserved words called modifiers that specify the properties of the data, methods, and classes and how they can be used.
- Examples of modifiers are public and static.
- Other modifiers are private, final, abstract, and protected.
  - A public datum, method, or class can be accessed by other programs.
  - A private datum or method cannot be accessed by other programs. Modifiers are discussed later in the course



- Vcvcvcv
- Vcvc
- Vc

Statements

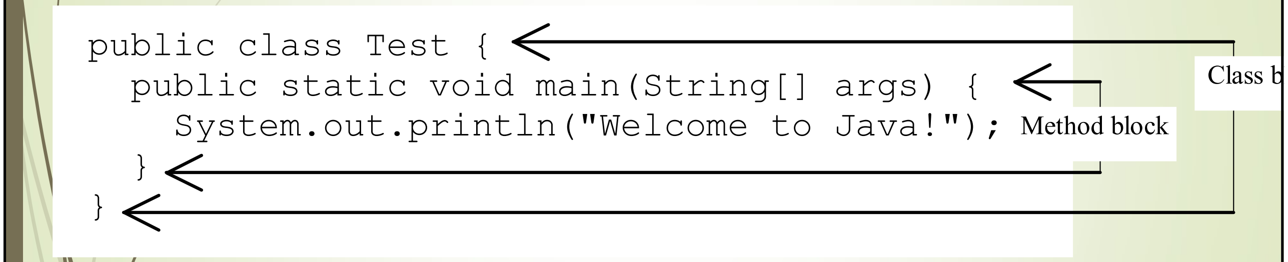
- A statement represents an action or a sequence of actions.

The statement System.out.println("Welcome to Java!"); in the exercise is a statement to display the greeting "Welcome to Java!“

- Every statement in Java ends with a semicolon (;)

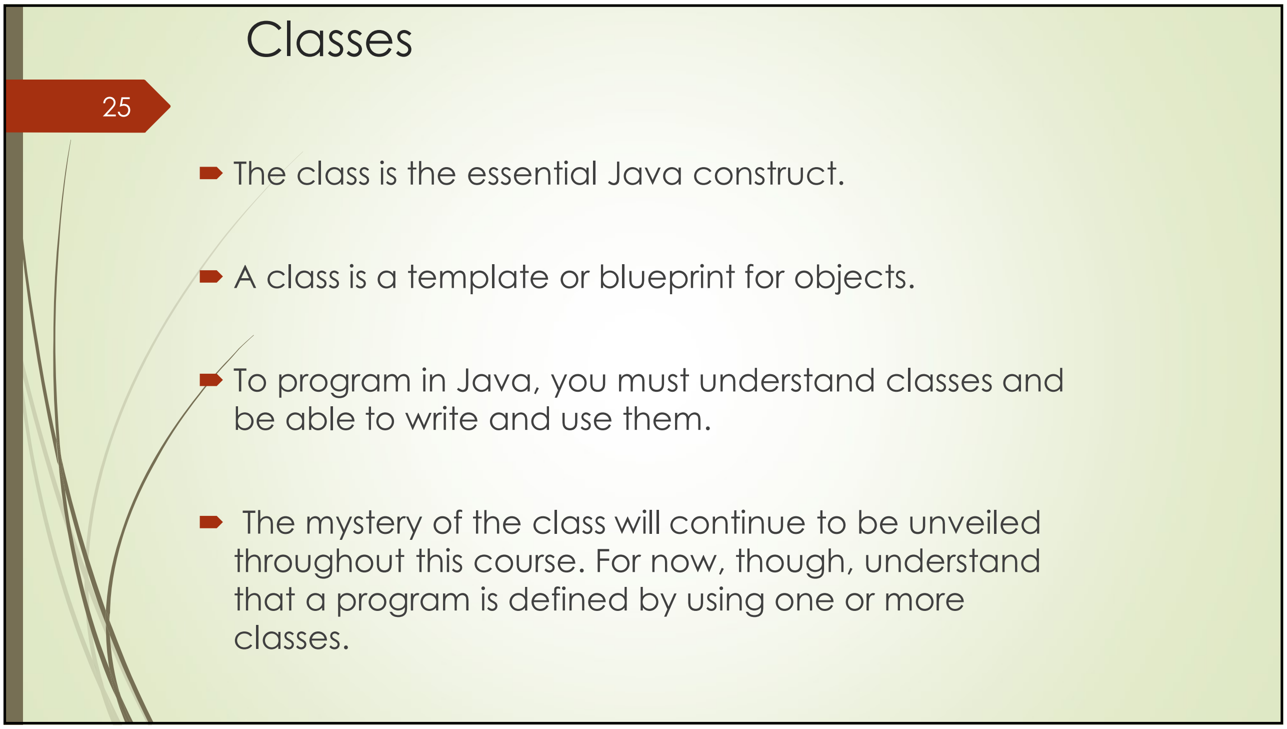
Blocks

- A pair of braces in a program forms a block that groups components of a program.



```
public class Test {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

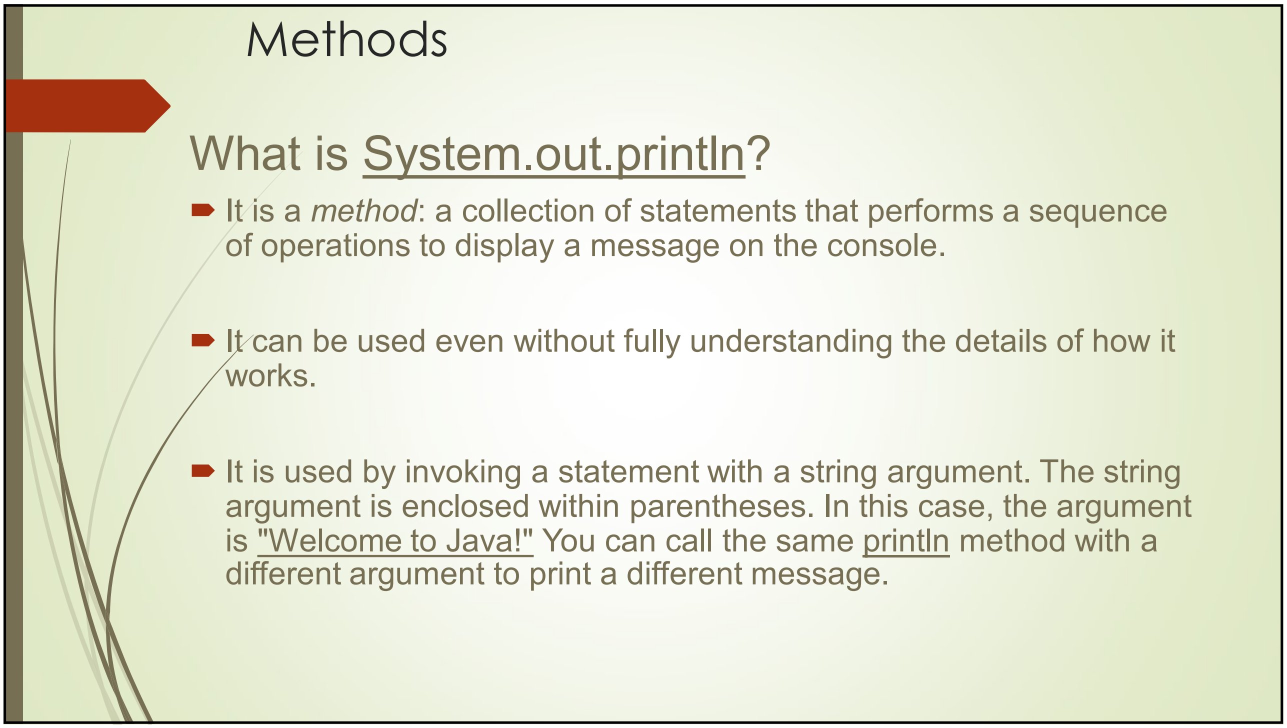
- All statements within a block are executed sequentially



Classes

25

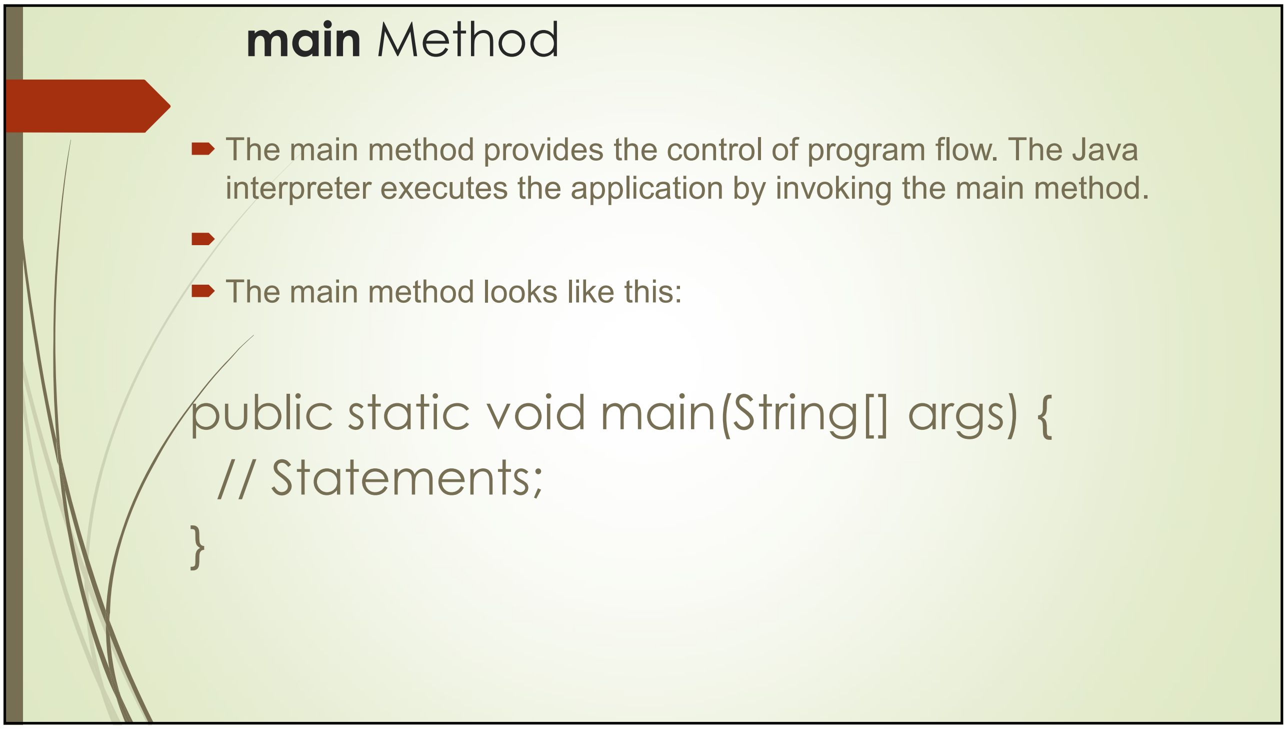
- The class is the essential Java construct.
- A class is a template or blueprint for objects.
- To program in Java, you must understand classes and be able to write and use them.
- The mystery of the class will continue to be unveiled throughout this course. For now, though, understand that a program is defined by using one or more classes.



Methods

What is System.out.println?

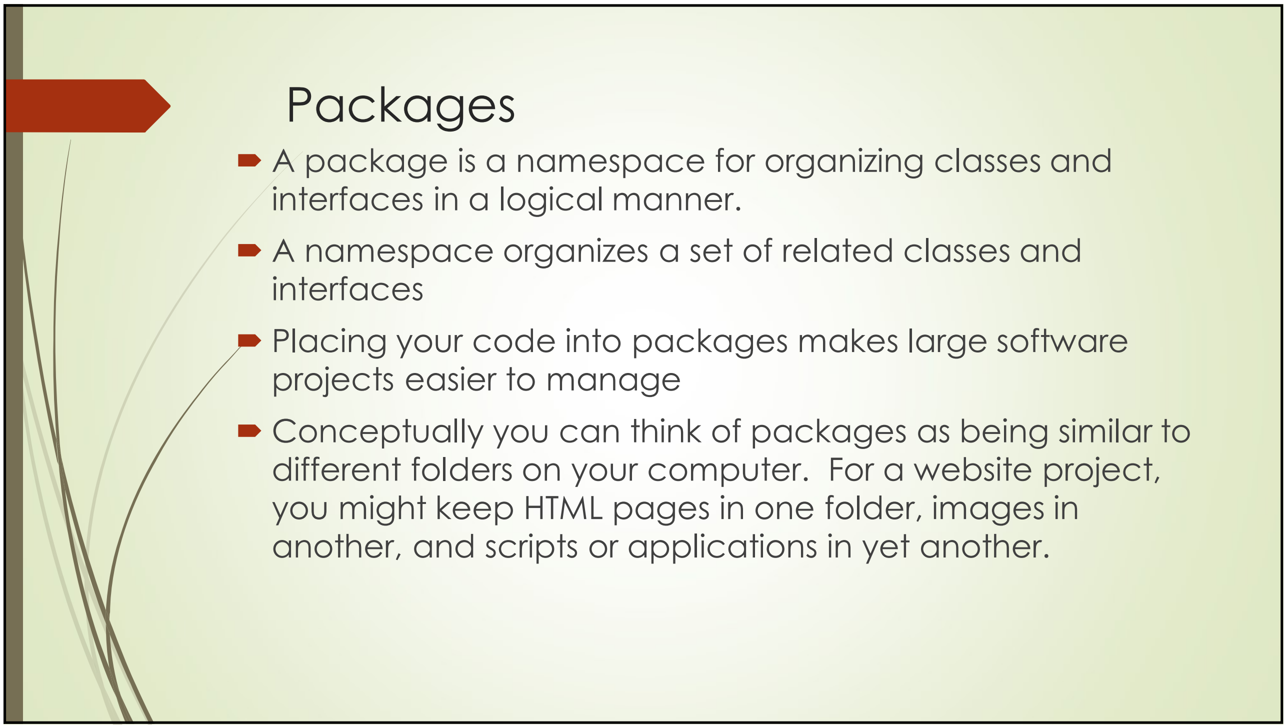
- It is a *method*: a collection of statements that performs a sequence of operations to display a message on the console.
- It can be used even without fully understanding the details of how it works.
- It is used by invoking a statement with a string argument. The string argument is enclosed within parentheses. In this case, the argument is "Welcome to Java!" You can call the same println method with a different argument to print a different message.



main Method

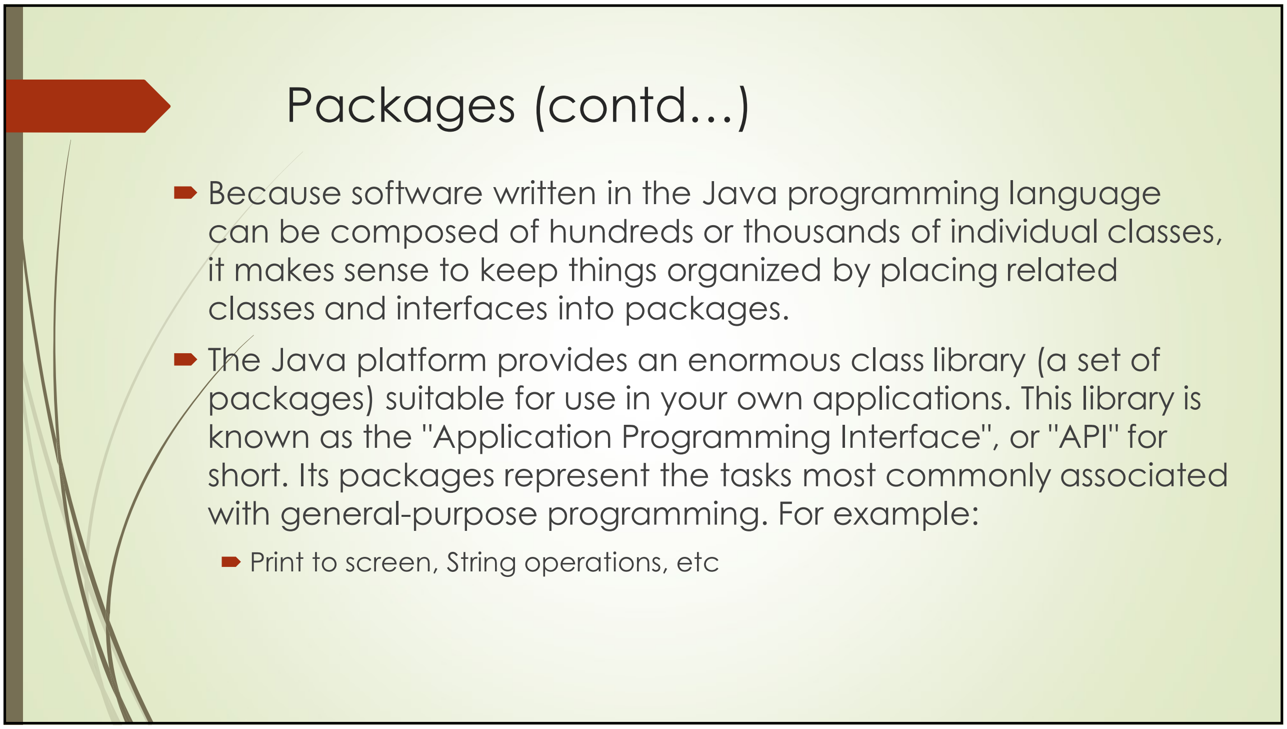
- The main method provides the control of program flow. The Java interpreter executes the application by invoking the main method.
-
- The main method looks like this:

```
public static void main(String[] args) {
  // Statements;
}
```



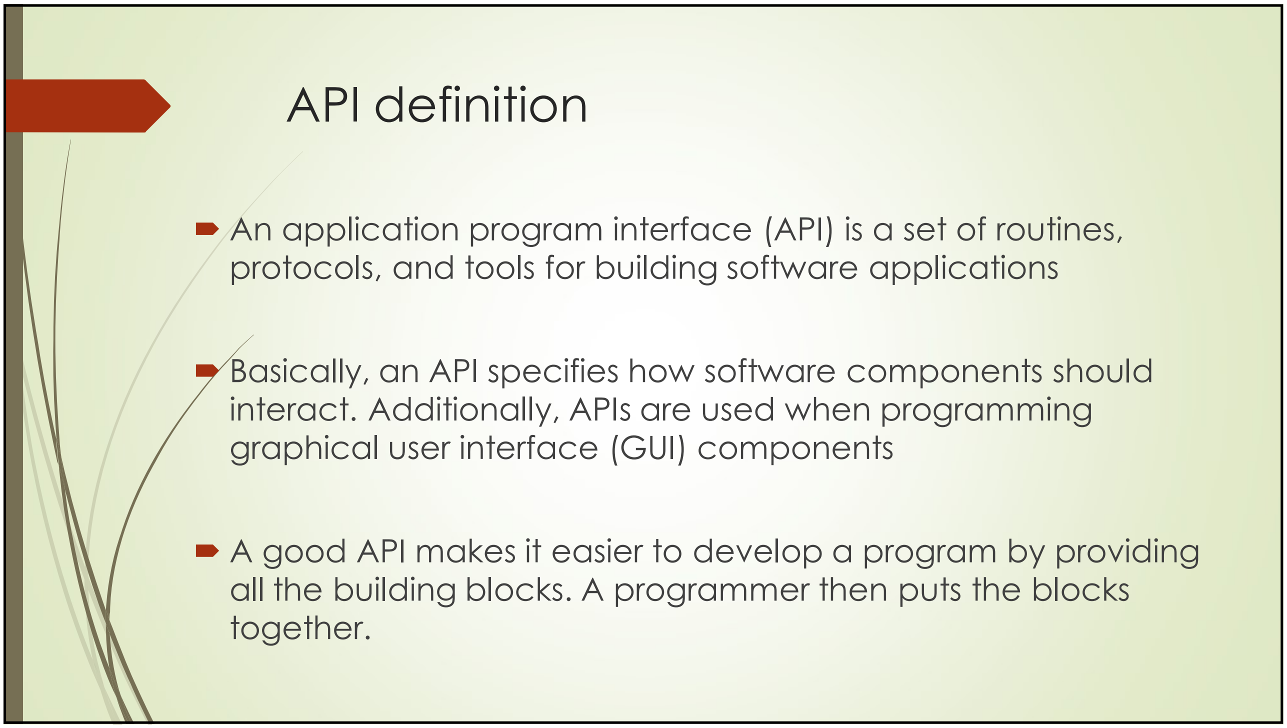
Packages

- A package is a namespace for organizing classes and interfaces in a logical manner.
- A namespace organizes a set of related classes and interfaces
- Placing your code into packages makes large software projects easier to manage
- Conceptually you can think of packages as being similar to different folders on your computer. For a website project, you might keep HTML pages in one folder, images in another, and scripts or applications in yet another.



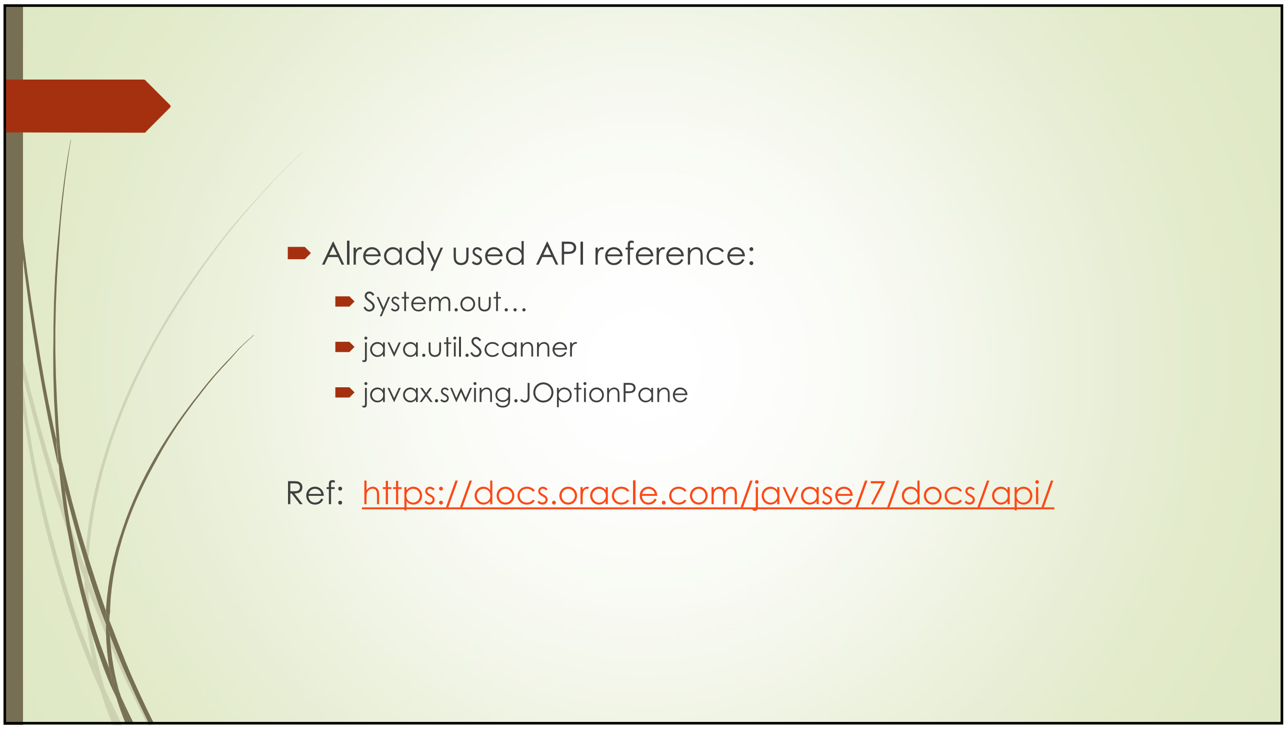
Packages (contd…)

- Because software written in the Java programming language can be composed of hundreds or thousands of individual classes, it makes sense to keep things organized by placing related classes and interfaces into packages.
- The Java platform provides an enormous class library (a set of packages) suitable for use in your own applications. This library is known as the "Application Programming Interface", or "API" for short. Its packages represent the tasks most commonly associated with general-purpose programming. For example:
  - Print to screen, String operations, etc



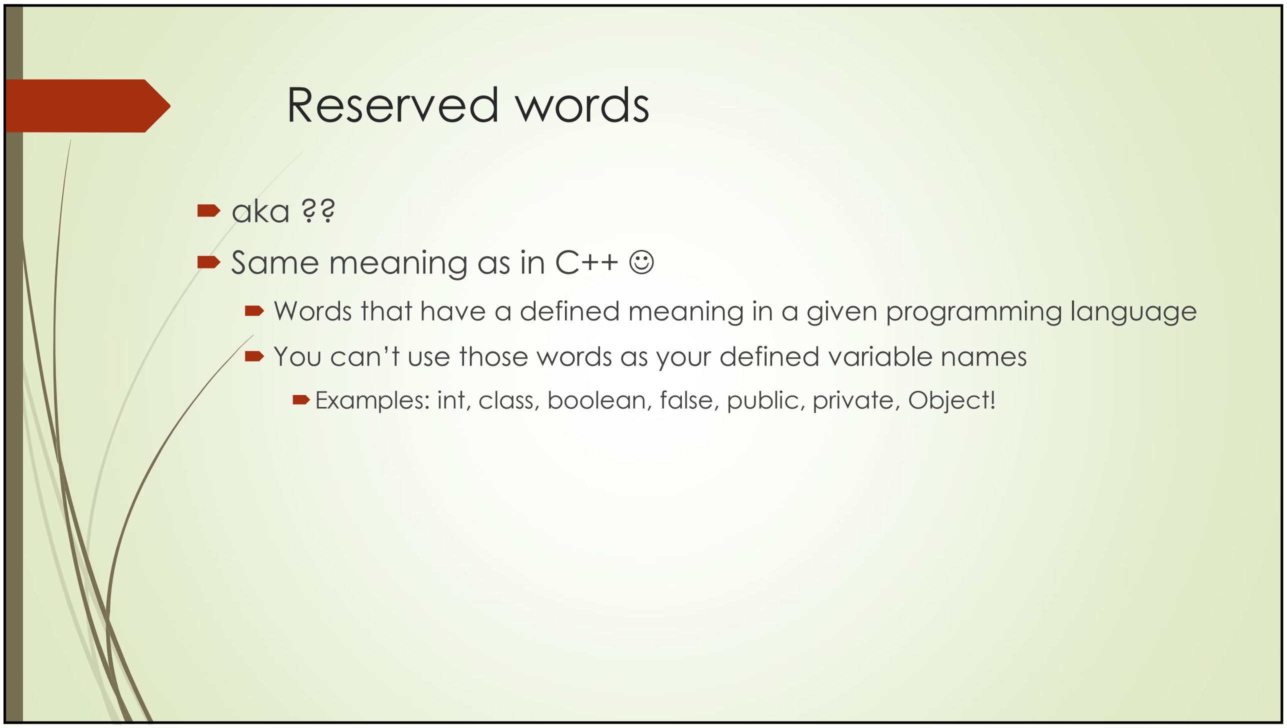
API definition

- An application program interface (API) is a set of routines, protocols, and tools for building software applications
- Basically, an API specifies how software components should interact. Additionally, APIs are used when programming graphical user interface (GUI) components
- A good API makes it easier to develop a program by providing all the building blocks. A programmer then puts the blocks together.



- Already used API reference:
  - System.out…
  - java.util.Scanner
  - javax.swing.JOptionPane

Ref: https://docs.oracle.com/javase/7/docs/api/



Reserved words

- aka ??
- Same meaning as in C++ ☺
  - Words that have a defined meaning in a given programming language
  - You can’t use those words as your defined variable names
    - Examples: int, class, boolean, false, public, private, Object!

Reflection: Anatomy of a Java Program

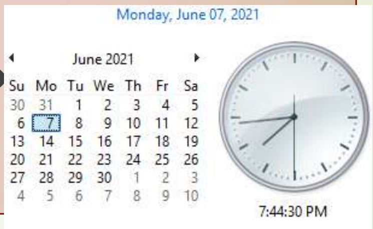
- Ref: the “TU-K Gate Management System”
  - The application to manage visits in a university
  - Model Classes
  - Model the flow (flow chart)
  - Code (only to an extent)

Thanks!

Your project!

Reflection Example: Anatomy of a Java Program

- Ref: the “Date” Assignment
  - The application requires the user to enter a date in June 2021, your program displays the day of the week.
  - E.g.
    - Input 1, output: Tuesday
    - Input : 7, output : Monday ➔
    - Input: 24, output : Thursday

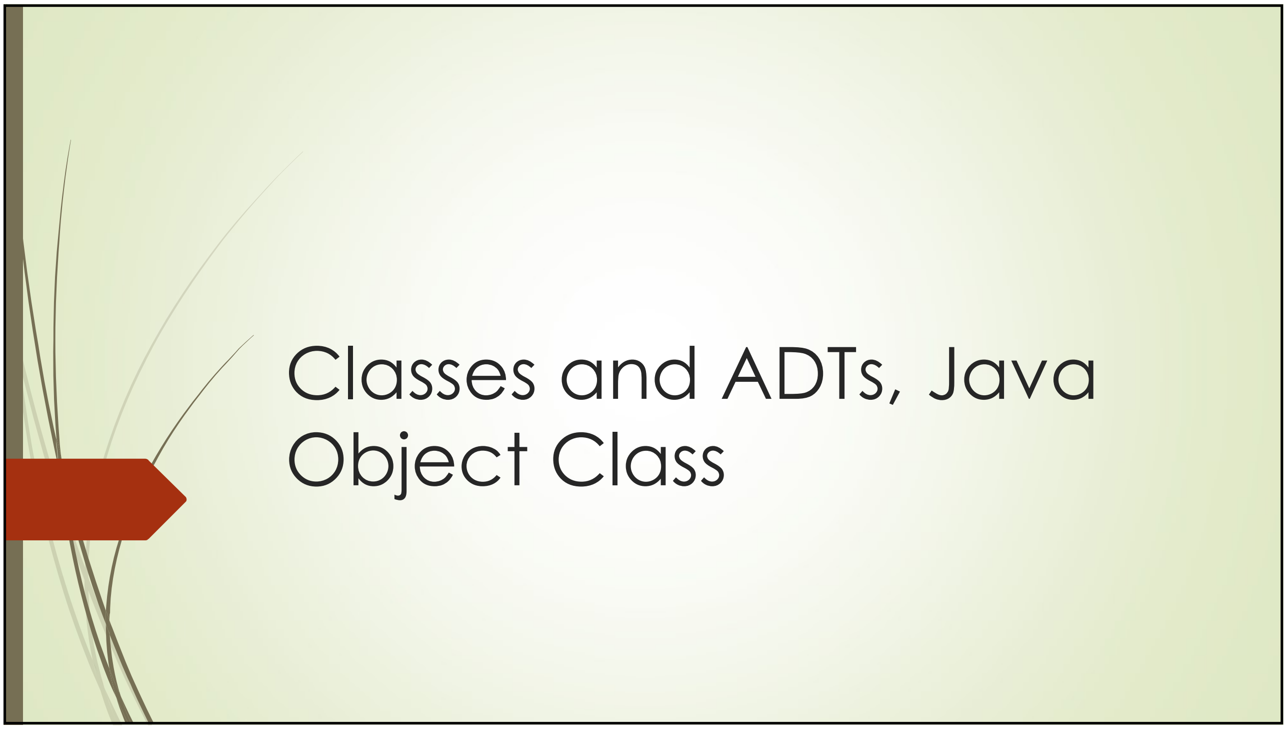


Monday, June 07, 2021

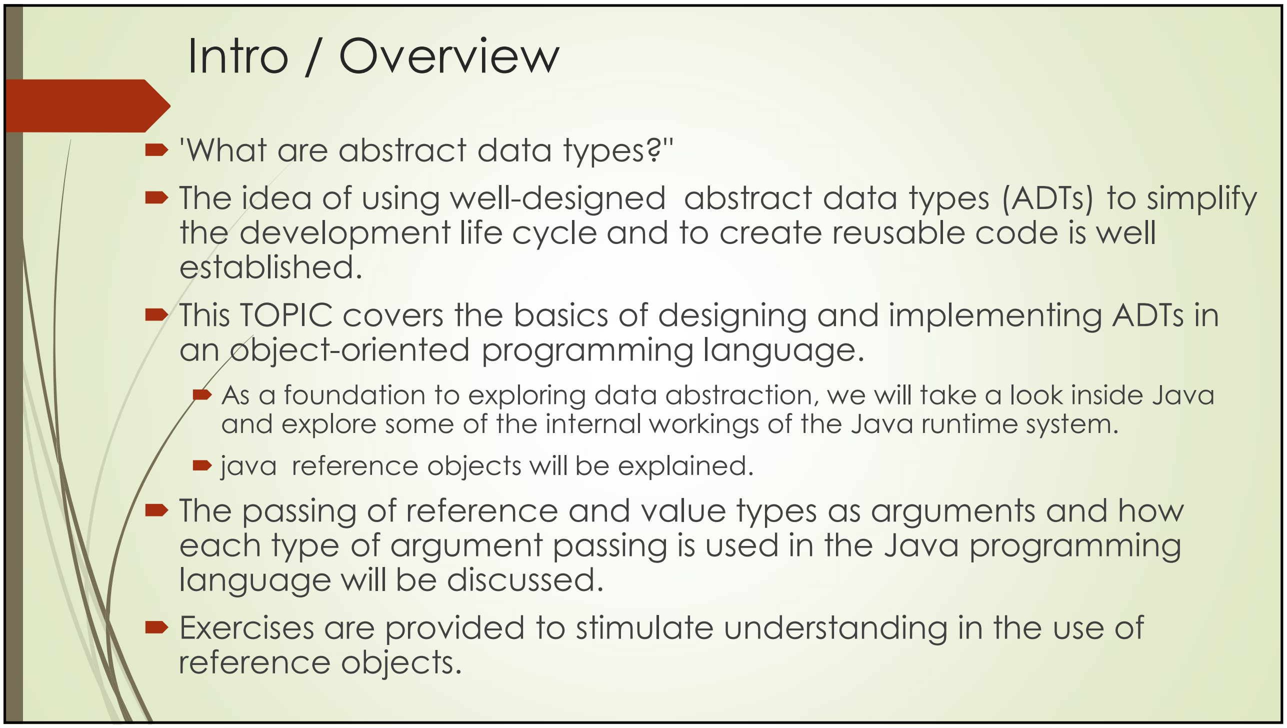
June 2021

Su	Mo	Tu	We	Th	Fr	Sa
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	1	2	3
4	5	6	7	8	9	10

7:44:30 PM

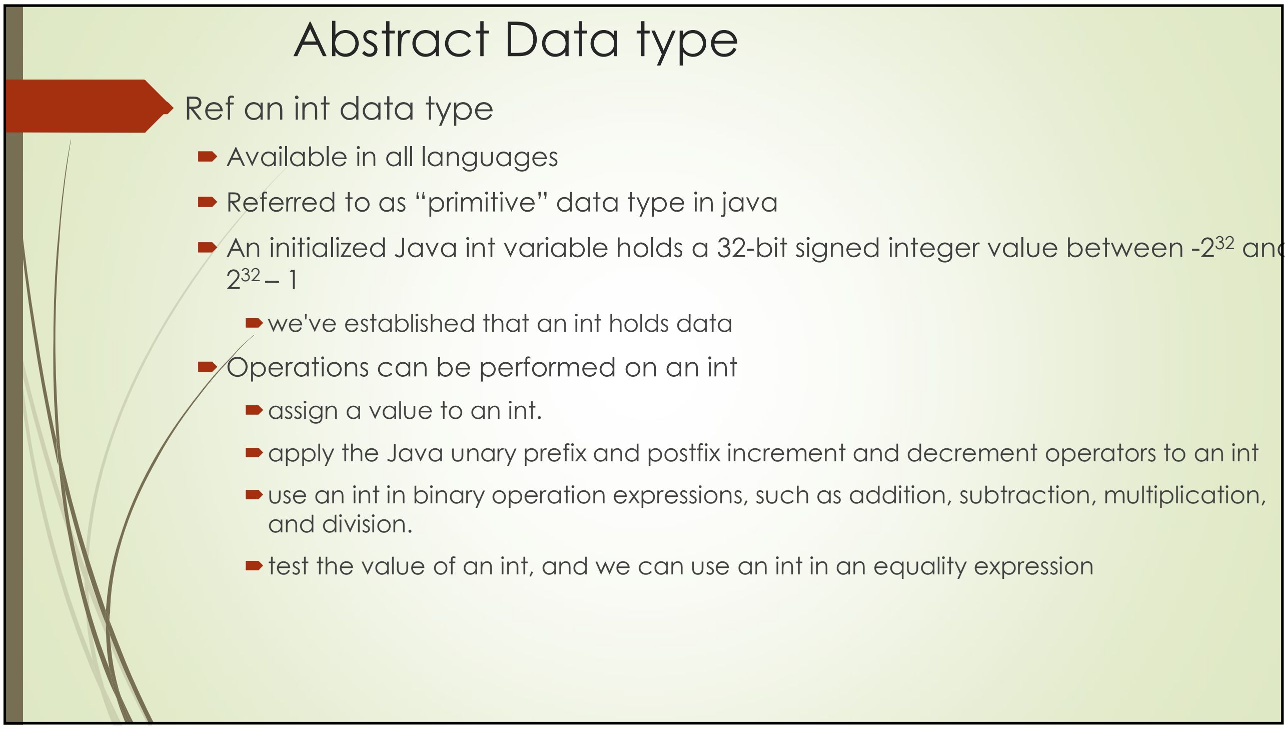


Classes and ADTs, Java Object Class



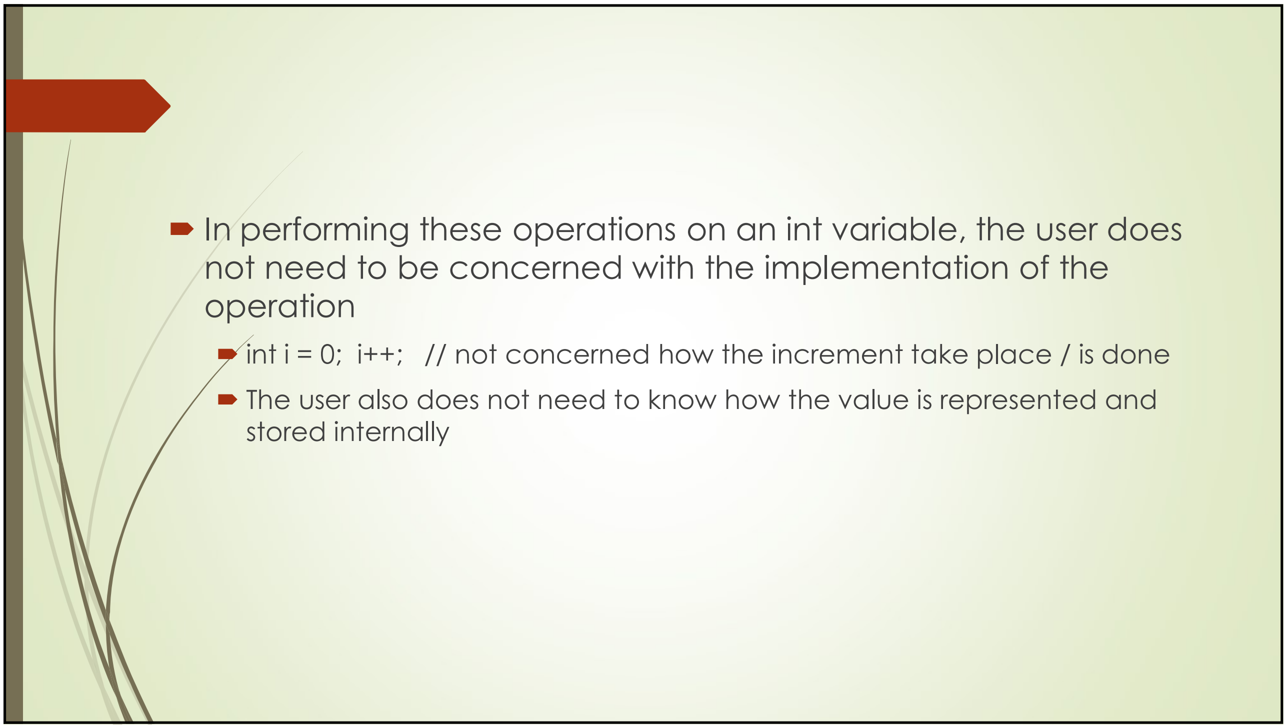
Intro / Overview

- 'What are abstract data types?"
- The idea of using well-designed abstract data types (ADTs) to simplify the development life cycle and to create reusable code is well established.
- This TOPIC covers the basics of designing and implementing ADTs in an object-oriented programming language.
  - As a foundation to exploring data abstraction, we will take a look inside Java and explore some of the internal workings of the Java runtime system.
  - java reference objects will be explained.
- The passing of reference and value types as arguments and how each type of argument passing is used in the Java programming language will be discussed.
- Exercises are provided to stimulate understanding in the use of reference objects.

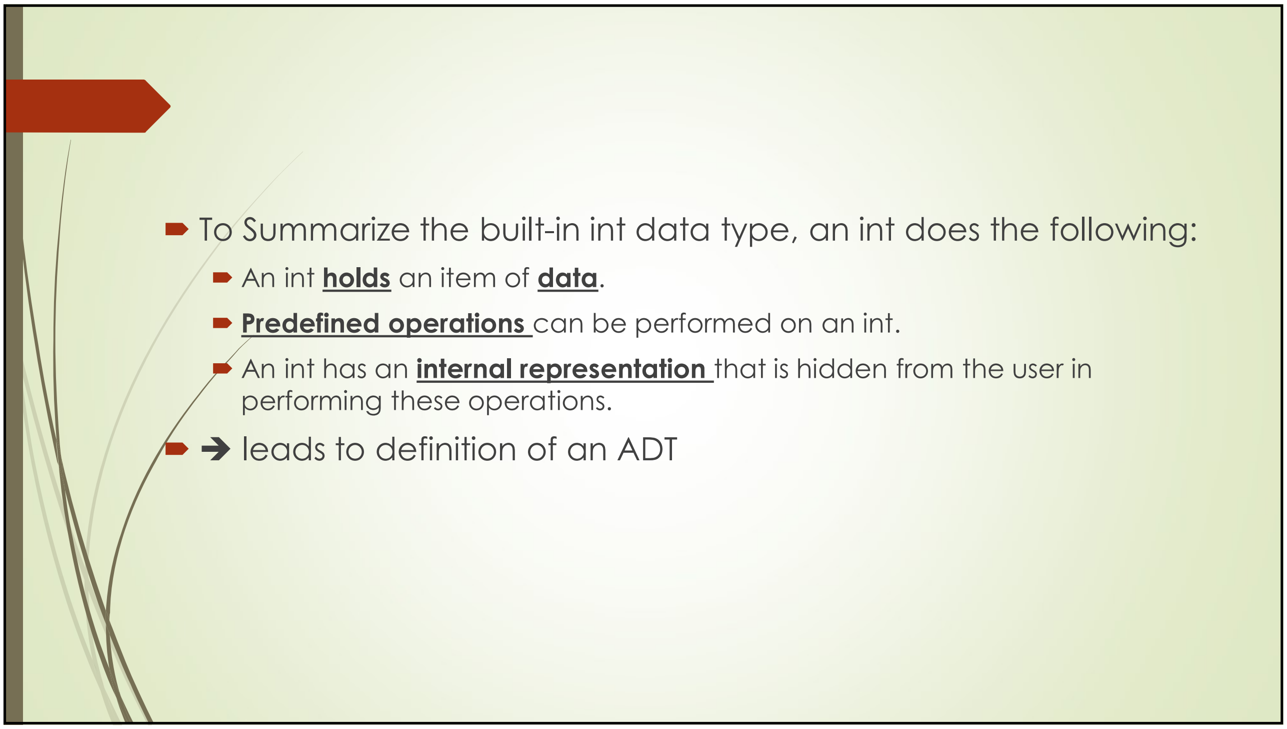


Abstract Data type

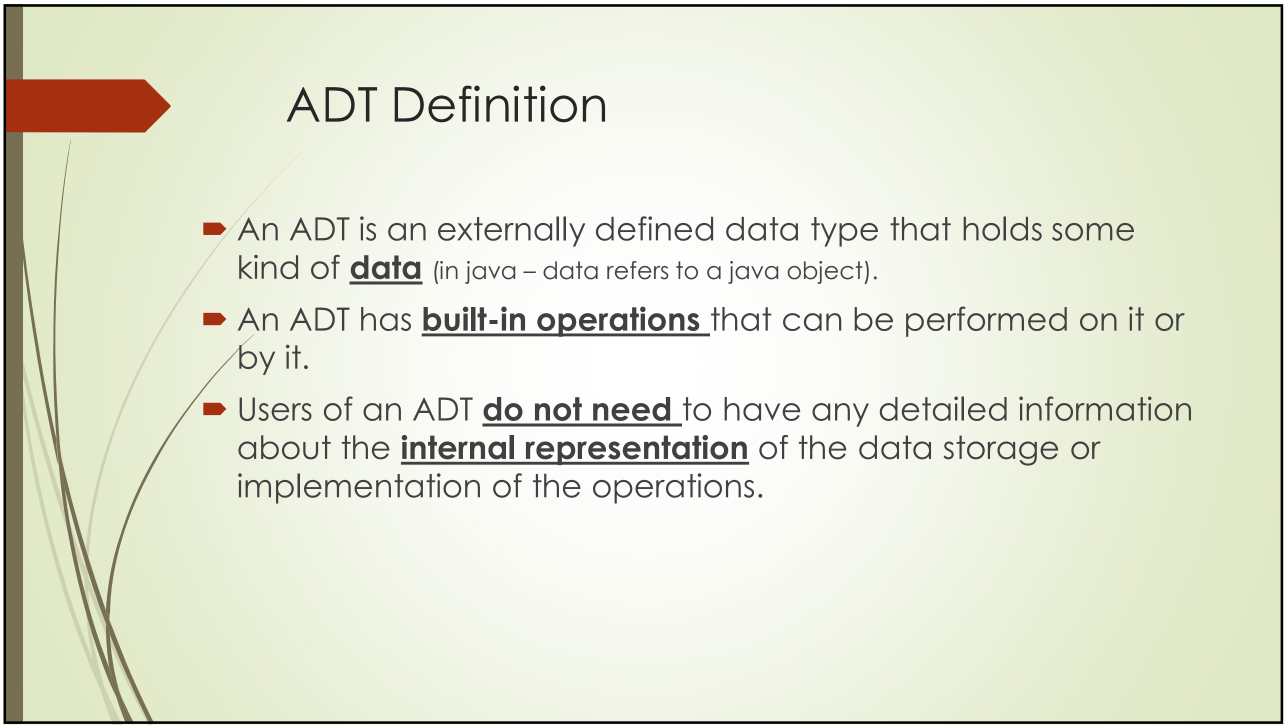
- Ref an int data type
  - Available in all languages
  - Referred to as “primitive” data type in java
  - An initialized Java int variable holds a 32-bit signed integer value between -2^{32} and $2^{32} – 1$
    - we've established that an int holds data
  - Operations can be performed on an int
    - assign a value to an int.
    - apply the Java unary prefix and postfix increment and decrement operators to an int
    - use an int in binary operation expressions, such as addition, subtraction, multiplication, and division.
    - test the value of an int, and we can use an int in an equality expression



- In performing these operations on an int variable, the user does not need to be concerned with the implementation of the operation
  - int i = 0; i++;   // not concerned how the increment take place / is done
  - The user also does not need to know how the value is represented and stored internally

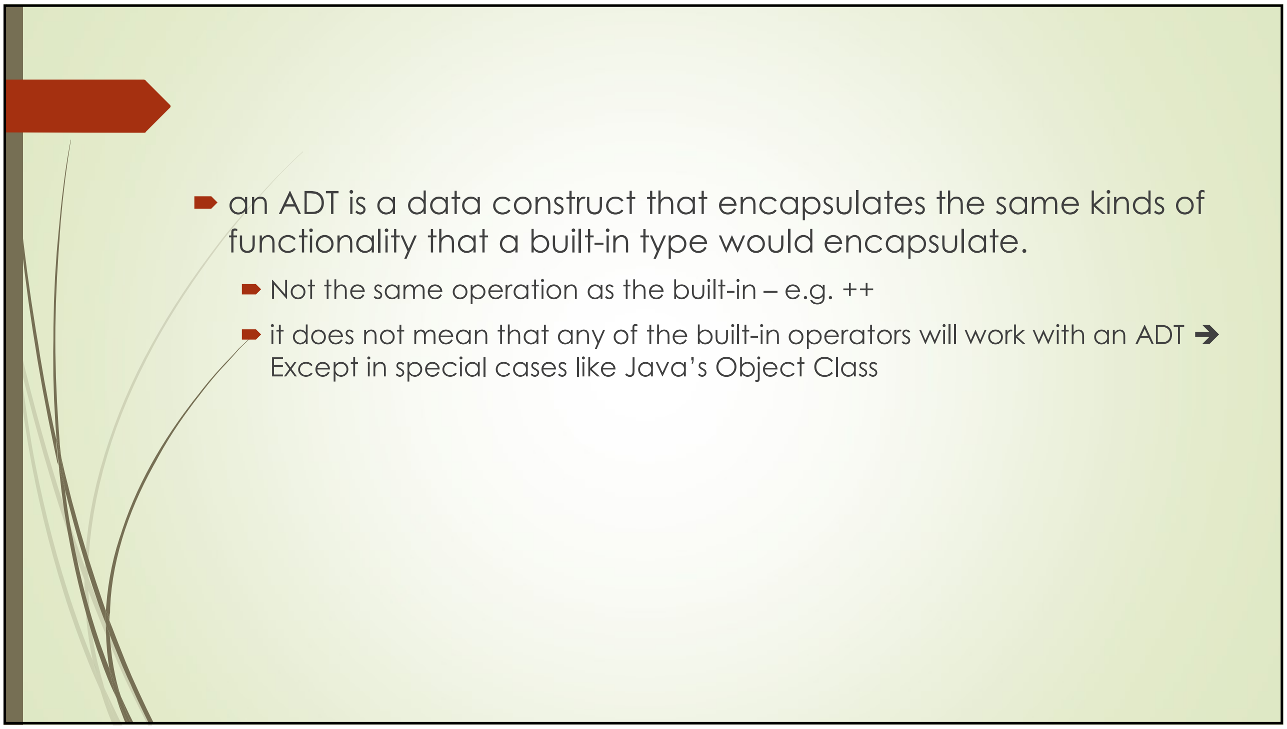


- To Summarize the built-in int data type, an int does the following:
  - An int **holds** an item of **data**.
  - **Predefined operations** can be performed on an int.
  - An int has an **internal representation** that is hidden from the user in performing these operations.
- ➔ leads to definition of an ADT

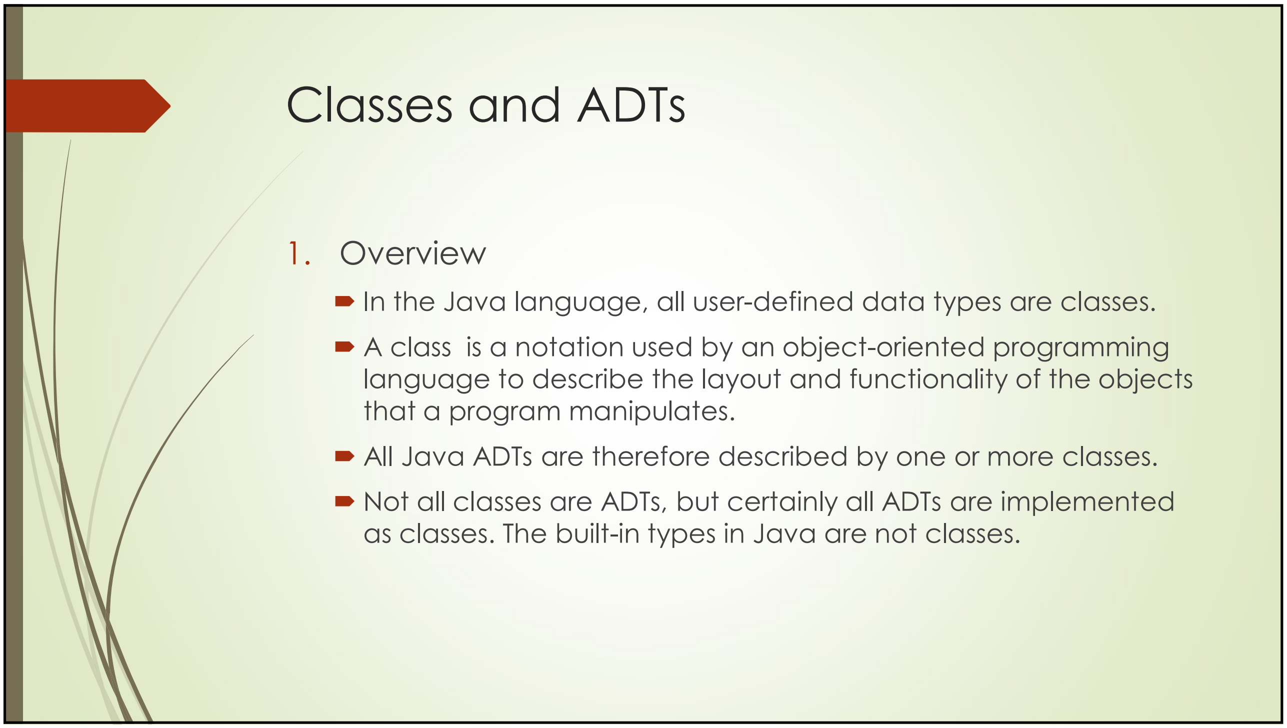


ADT Definition

- An ADT is an externally defined data type that holds some kind of **data** (in java – data refers to a java object).
- An ADT has **built-in operations** that can be performed on it or by it.
- Users of an ADT **do not need** to have any detailed information about the **internal representation** of the data storage or implementation of the operations.



- an ADT is a data construct that encapsulates the same kinds of functionality that a built-in type would encapsulate.
  - Not the same operation as the built-in – e.g. ++
  - it does not mean that any of the built-in operators will work with an ADT ➔ Except in special cases like Java's Object Class



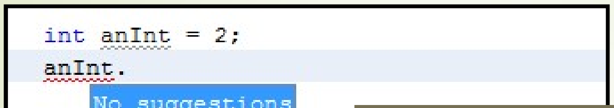
Classes and ADTs

1. Overview
   - In the Java language, all user-defined data types are classes.
   - A class is a notation used by an object-oriented programming language to describe the layout and functionality of the objects that a program manipulates.
   - All Java ADTs are therefore described by one or more classes.
   - Not all classes are ADTs, but certainly all ADTs are implemented as classes. The built-in types in Java are not classes.

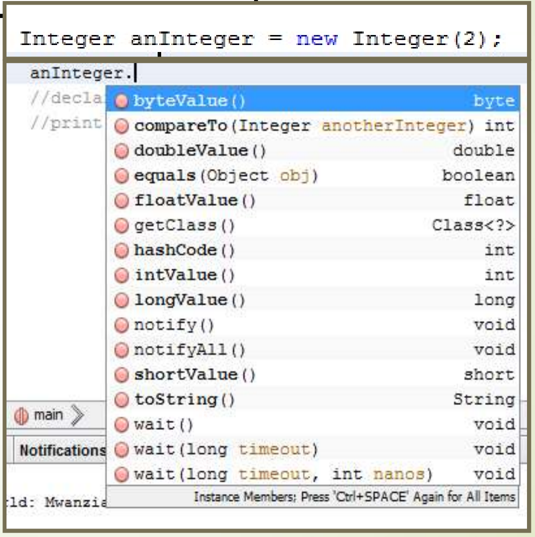
Classes and ADTs

2. Reference Objects and Value Types
   - In Java, two basic types of variables exist: primitive types and reference types
   - Primitive types - the standard built-in types we would expect to find in any modern programming language: int, long, short, byte, char, boolean, float, double, void (special)
   - **Reference types** - any variables that refer to an object.
   - This is an important distinction, because the two variable types are treated differently in various situations.
     - All reference type objects are of a specific class

- Compare and Contrast the following:



```
int anInt = 2;
anInt.
     No suggestions
```



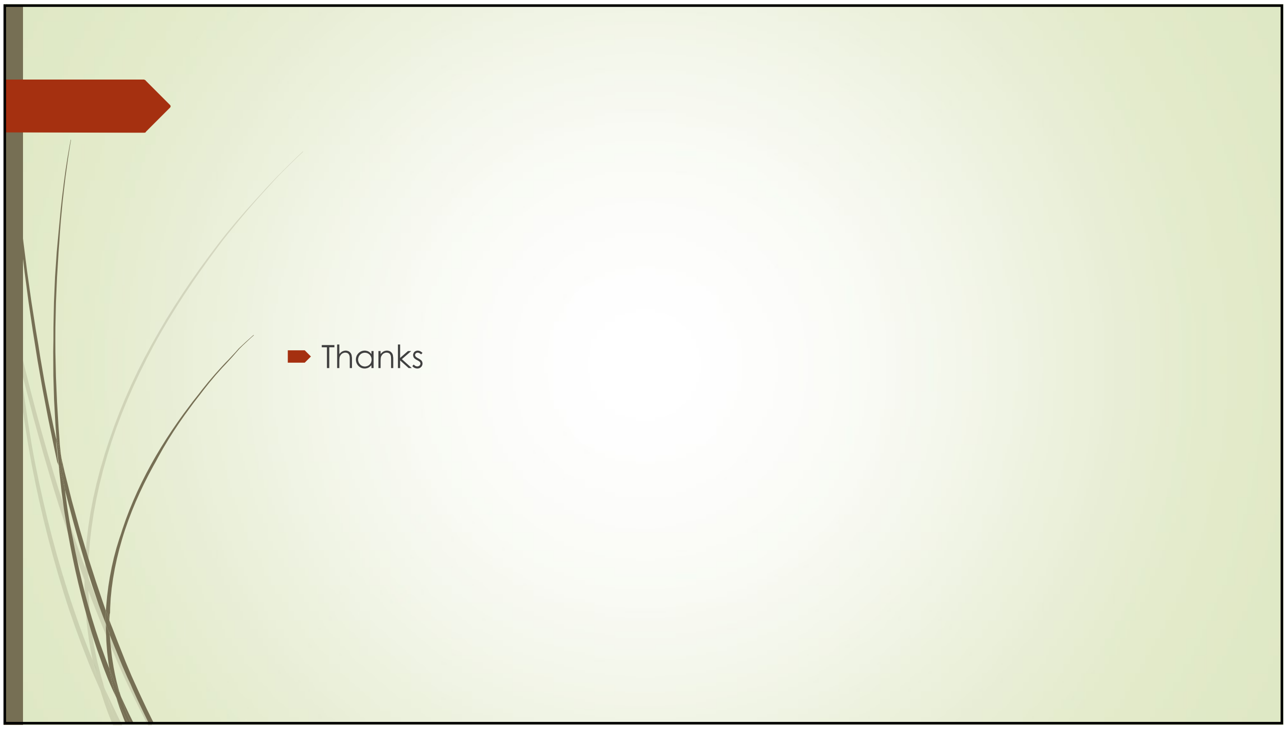
```
Integer anInteger = new Integer(2);
anInteger.
//decla  byteValue()                          byte
//print  compareTo(Integer anotherInteger)    int
         doubleValue()                        double
         equals(Object obj)                   boolean
         floatValue()                         float
         getClass()                           Class<?>
         hashCode()                           int
         intValue()                           int
         longValue()                          long
         notify()                             void
         notifyAll()                          void
         shortValue()                         short
         toString()                           String
         wait()                               void
         wait(long timeout)                   void
         wait(long timeout, int nanos)        void
         Instance Members; Press 'Ctrl+SPACE' Again for All Items
```

3. The Object Class

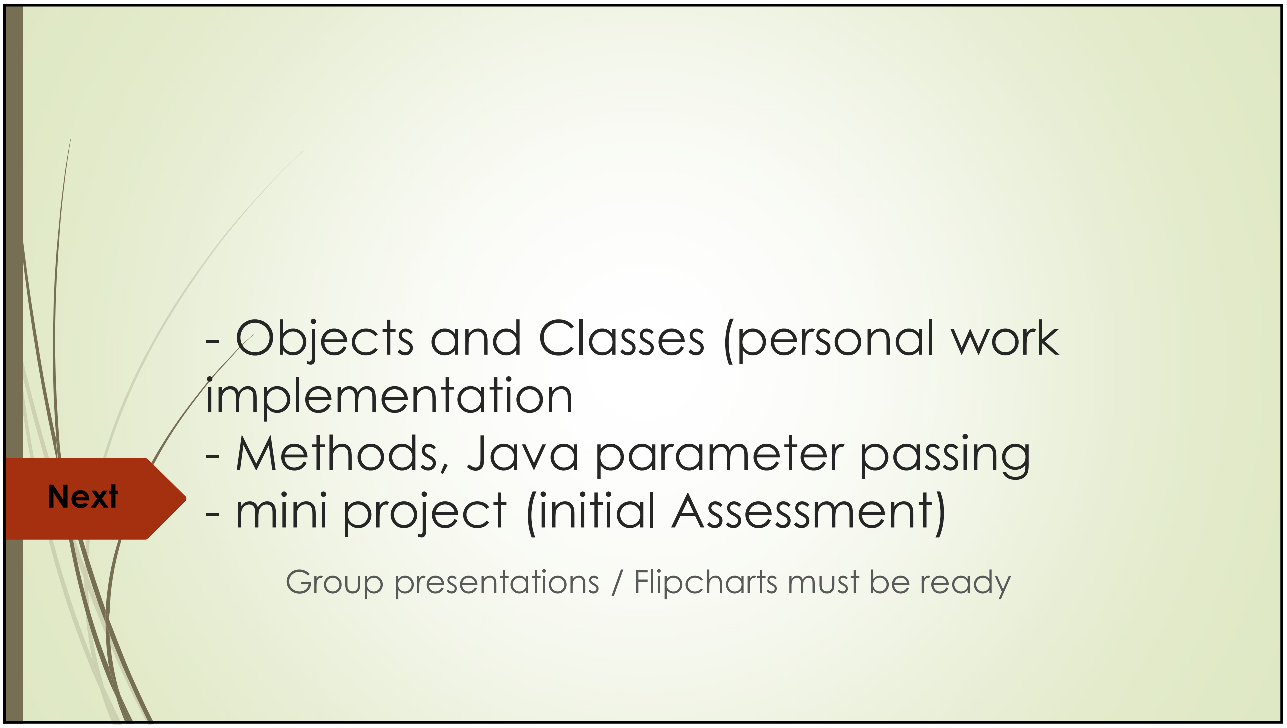
- All classes in Java are derived from the root class Object
- So, given the rules of inheritance and polymorphism, an Object class variable can refer to any reference object of any class.
- One of Java's strengths is the fact that it uses a polymorphic model wherein all classes are derived from a common root.
- All objects share a common base Application Programming Interface (API)

4. The Object Class (Wrapper classes)

- The Java Development Kit (JDK) comes complete with a set of classes defined as the core API.
- All these classes are the Java equivalent of a standard library. The Java core classes include wrapper classes for all the primitive types:
  - Integer for int,
  - Double for double,
  - Float for float, etc.
- Of course, all these wrapper classes are derived from the root class Object as well.



- Thanks



Next

- Objects and Classes (personal work implementation
- Methods, Java parameter passing
- mini project (initial Assessment)

Group presentations / Flipcharts must be ready