Basics of Programming in Java

Outline

- Java Architecture, Class paths, Sample Program
- Classes, Objects, Constructors
- Packages and Data Types
- Conditional Statements
- Access Modifiers
- Exception Handling
- Java Collections

Java

- Java is a multi-platform
- Java is object-oriented
- Java is network-centric language that can be used as a platform in itself.
- It is a fast, secure, reliable programming language for coding everything from mobile apps and enterprise software to big data applications and server-side technologies.

Java as programming tools

- Java is a **high-level programming language** developed by James Gosling and his team at Sun Microsystems (now part of Oracle Corporation).
- It was **first released in 1995** and has since become one of the most popular programming languages in the world.
- Java is **known for its platform independence,** meaning that Java code can run on any computer or device that has a **Java Virtual Machine (JVM)** installed.
- Some benefits of Java include its simplicity, object-oriented nature, robustness, security, and large community support.
- Java is used for **developing a wide range of applications**, including web and mobile applications, desktop software, games, and more.

Benefits of Java

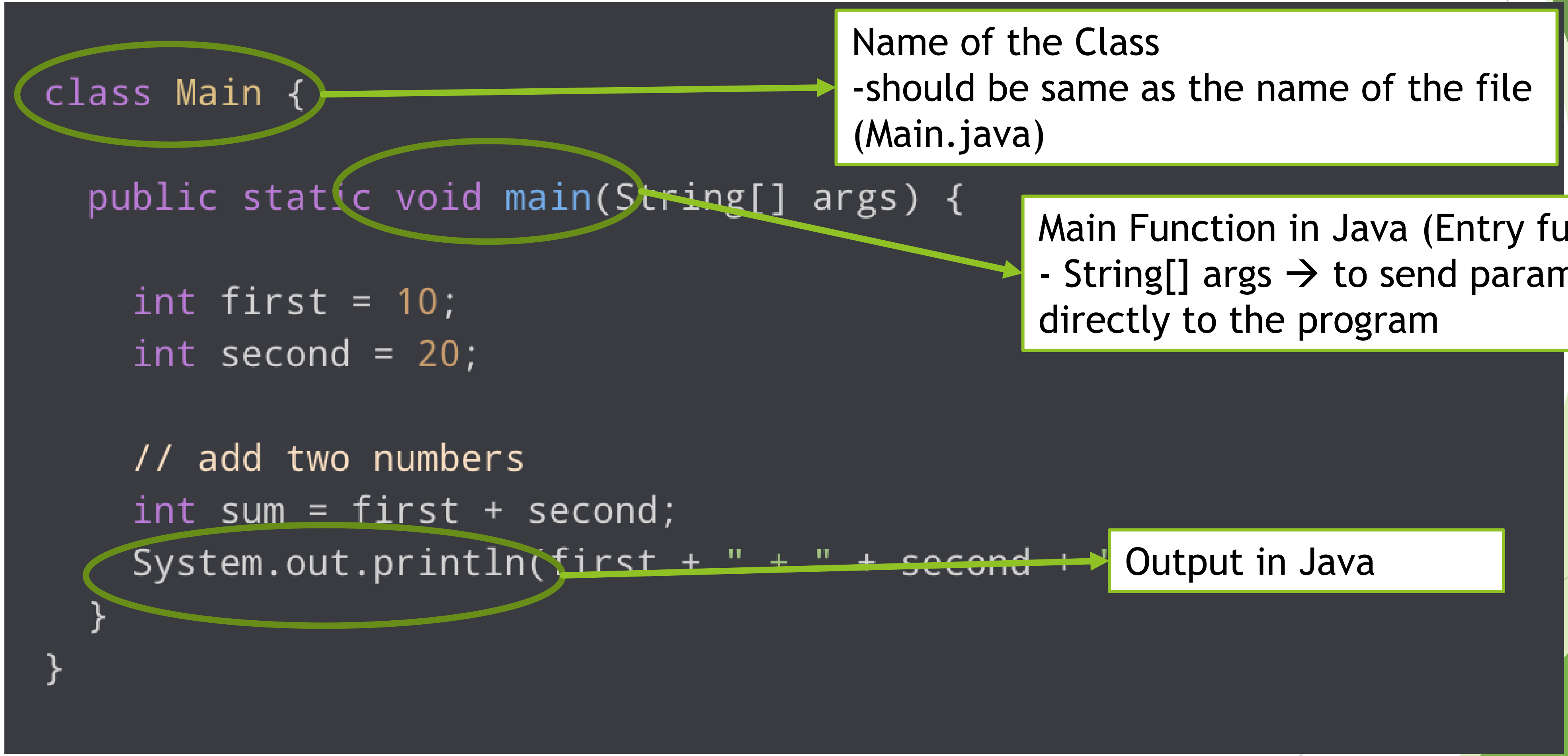
- Platform Independence: Java is known for its "write once, run anywhere" capability. This means that Java code can be written on one platform (such as Windows) and run on any other platform (such as Linux or Mac) that has a Java Virtual Machine (JVM) installed. This platform independence makes Java highly portable and flexible.
- Object-Oriented Programming (OOP): Java is designed as an object-oriented programming language, which promotes the concept of objects and classes. This approach allows for modular and reusable code, making it easier to manage and maintain large-scale applications. OOP concepts such as encapsulation, inheritance, and polymorphism are supported in Java.
- Robust and Secure: Java is known for its robustness and reliability. It provides features like automatic memory management (garbage collection) and exception handling, which help prevent common programming errors and enhance the stability of Java applications. Additionally, Java has built-in security features, such as a security manager and sandbox model, which make it suitable for developing secure applications.

- Rich API and Large Community Support: Java has an extensive Application Programming Interface (API) that provides ready-to-use classes and libraries for various purposes, such as networking, file handling, user interface development, and more. The Java API simplifies application development by providing pre-built components. Furthermore, Java has a large and active community of developers who contribute to its ecosystem, offering support, libraries, frameworks, and tools.
- Multithreading and Concurrency: Java supports multithreading, allowing developers to write programs that can execute multiple threads simultaneously. This feature is beneficial for applications that require concurrent execution, such as web servers or applications with a graphical user interface. Java provides built-in libraries and mechanisms for handling threads and synchronization.
- Performance and Efficiency: Java has a Just-In-Time (JIT) compiler that dynamically compiles Java bytecode into native machine code at runtime. This compilation process optimizes the performance of Java applications, making them faster and more efficient. Additionally, Java's garbage collector manages memory efficiently, automatically deallocating memory that is no longer in use.
- Wide Range of Applications: Java is a versatile language used for developing a wide range of applications, from web and enterprise applications to mobile apps, desktop software, scientific applications, and embedded systems. It is the language behind popular frameworks like Spring and technologies like Android development.

A Simple Java Program

- A Java program is composed of classes and methods. The program execution starts from the main method.
- Data types in Java include primitive types (e.g., int, double, boolean) and reference types (e.g., String, arrays, objects).
- Variables are used to store data. They must be declared with a specific data type and can be assigned values using the assignment operator (=).
- Initialization is the process of giving a variable an initial value.
- Operators are used to perform operations on data. Examples include arithmetic operators (+, -, *, /), relational operators (>, <, ==), and logical operators (&&, ||).
- Strings are used to represent sequences of characters. They are objects in Java and have various built-in methods for string manipulation.
- Control flow statements are used to control the execution flow of a program. Examples include if-else statements, loops (for, while, do-while), and switch statements

Examples



```
class Main {

  public static void main(String[] args) {

    int first = 10;
    int second = 20;

    // add two numbers
    int sum = first + second;
    System.out.println(first + " + " + second + "
  }
}
```

Name of the Class
-should be same as the name of the file (Main.java)

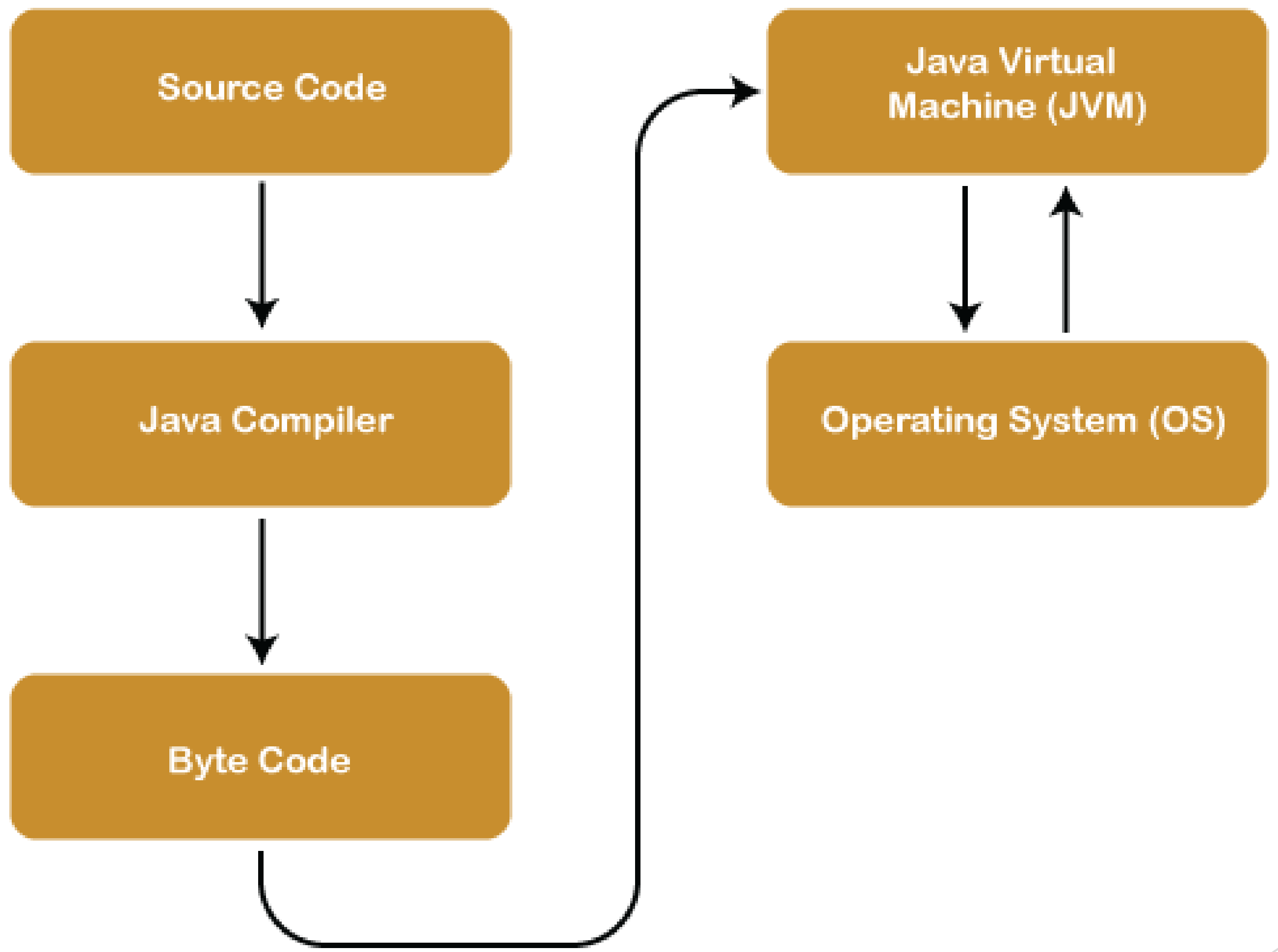
Main Function in Java (Entry function)
- String[] args → to send parameters directly to the program

Output in Java

Java Architecture

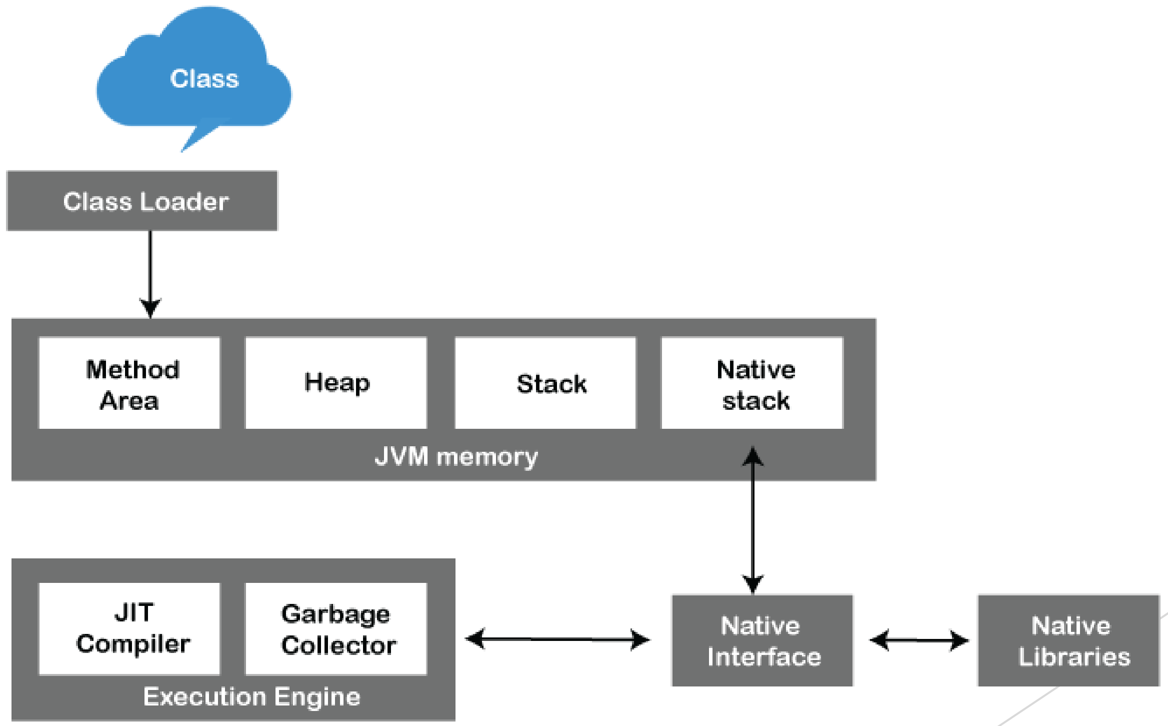
- Architecture is a collection of components, i.e., JVM, JRE, and JDK.
- It integrates the process of interpretation and compilation
- It defines all the processes involved in creating a Java program
- Java Architecture explains each and every step of how a program is compiled and executed.

Java Architecture (contd...)



Java Virtual Machine

- JVM is an abstract machine that provides the environment in which Java bytecode is executed.



Java Runtime Environment

- It provides an environment in which Java programs are executed
- JRE takes our Java code, integrates it with the required libraries, and then starts the JVM to execute it

Java Development Kit

- It is a software development environment used in the development of Java applications
- Java Development Kit holds JRE, a compiler, an interpreter or loader, and several development tools in it

Class and Objects

- Java is an object-oriented programming language.
- To create a class, use the keyword class:

```
public class Main {
  int x = 5;
}
```

- In Java, an object is created from a class. We have already created the class named Main, so now we can use this to create objects.

```
public class Main {
  int x = 5;

  public static void main(String[] args) {
    Main myObj = new Main();
    System.out.println(myObj.x);
  }
}
```

Constructor

- A constructor in Java is a special method that is used to initialize objects
- The constructor is called when an object of a class is created
- Constructors can also take parameters, which is used to initialize attributes

Constructor (contd...)

```
// Create a Main class
public class Main {
  int x;  // Create a class attribute

  // Create a class constructor for the Main class
  public Main() {
    x = 5;  // Set the initial value for the class attribute x
  }

  public static void main(String[] args) {
    Main myObj = new Main(); // Create an object of class Main (This will call the constructor)
    System.out.println(myObj.x); // Print the value of x
  }
}
```

Packages

- Packages help in organizing and managing large codebases, providing a way to create a modular and hierarchical structure for classes and preventing naming conflicts.
- A package is a way to organize related classes, interfaces, and sub-packages into a hierarchical structure.
- It provides a means of grouping and managing classes, avoiding naming conflicts, and promoting code modularity and reusability.

Data Types in Java

- Data types are divided into two groups
  - Primitive data types - includes byte, short, int, long, float, double, boolean and char
  - Non-primitive data types - such as String, Arrays and Classes
- User defined class can also be used as a data type in java.

- eg: `int numberOfTokens = 100;`
- eg: `String myString = “my_input_string”;`

Conditional Statements

```
if (condition1) {
  // block of code to be
} else if (condition2) {
  // block of code to be
} else {
  // block of code to be
}
```

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

```
 true

1 is false and condition2 is true

1 is false and condition2 is false
```

Access Protection Mechanism

- Java provides access modifiers to control the visibility and accessibility of classes, variables, and methods.
- The access modifiers are "public", "protected", "private", and the default (no modifier).
- "Public" allows access from anywhere, "protected" allows access within the same package or subclasses, "private" restricts access to within the class itself, and the default restricts access to the same package.
- Access protection mechanisms ensure encapsulation, data hiding, and proper separation of concerns in an object-oriented system.

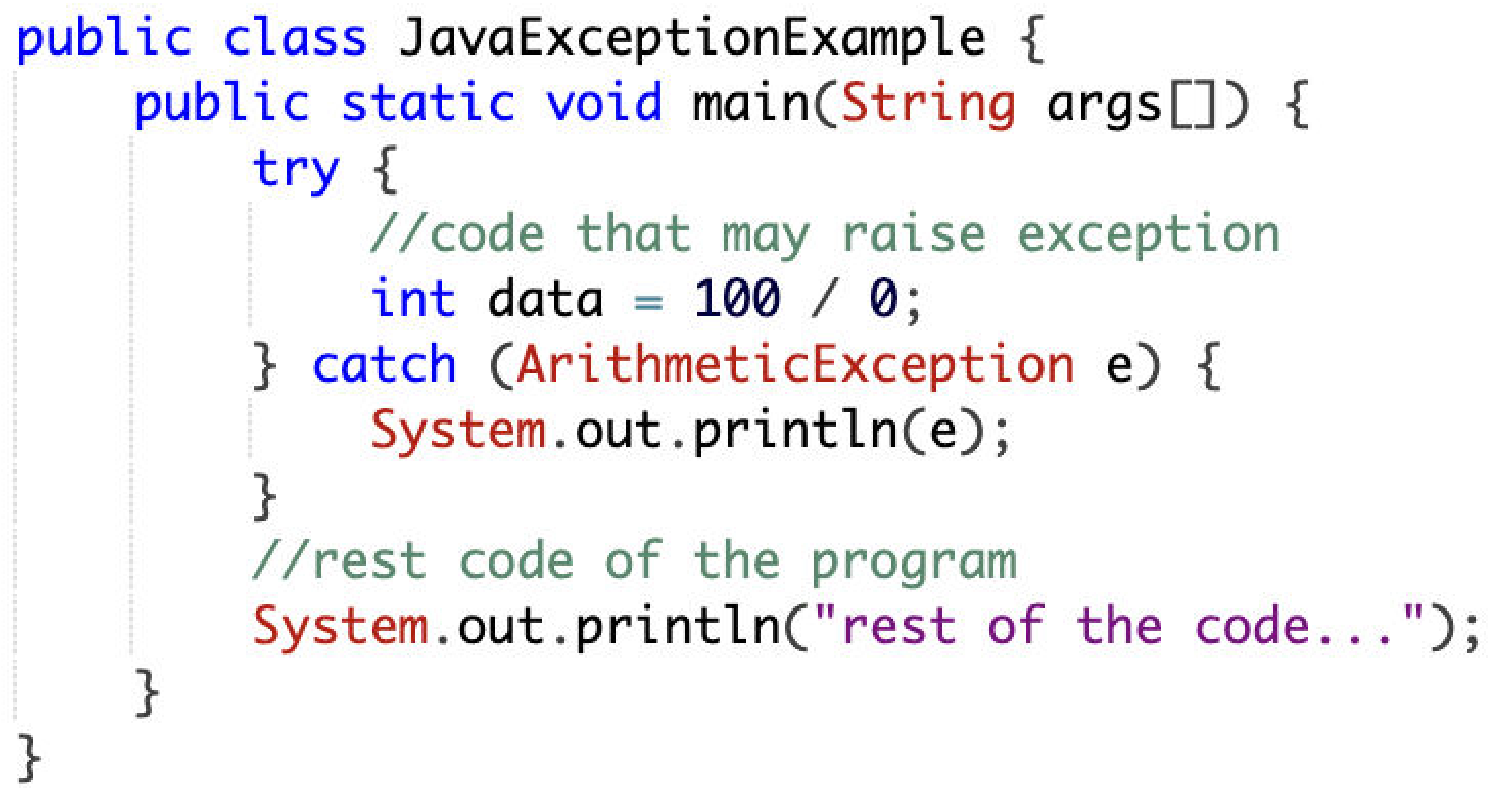
Exception Handling

- The Exception Handling in Java is one of the powerful mechanism to handle the runtime errors so that the normal flow of the application can be maintained.
- In Java, an exception is an event that disrupts the normal flow of the program
- It is an object which is thrown at runtime.
- The core advantage of exception handling is to maintain the normal flow of the application
- An exception normally disrupts the normal flow of the application; that is why we need to handle exceptions

Exception Handling (contd...)

Keyword	Description
try	The "try" keyword is used to specify a block where we should place an exception code. It means we can't use try block alone. The try block must be followed by either catch or finally.
catch	The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later.
finally	The "finally" block is used to execute the necessary code of the program. It is executed whether an exception is handled or not.
throw	The "throw" keyword is used to throw an exception.
throws	The "throws" keyword is used to declare exceptions. It specifies that there may occur an exception in the method. It doesn't throw an exception. It is always used with method signature.

Exception Handling (contd...)



```java
public class JavaExceptionExample {
    public static void main(String args[]) {
        try {
            //code that may raise exception
            int data = 100 / 0;
        } catch (ArithmeticException e) {
            System.out.println(e);
        }
        //rest code of the program
        System.out.println("rest of the code...");
    }
}
```

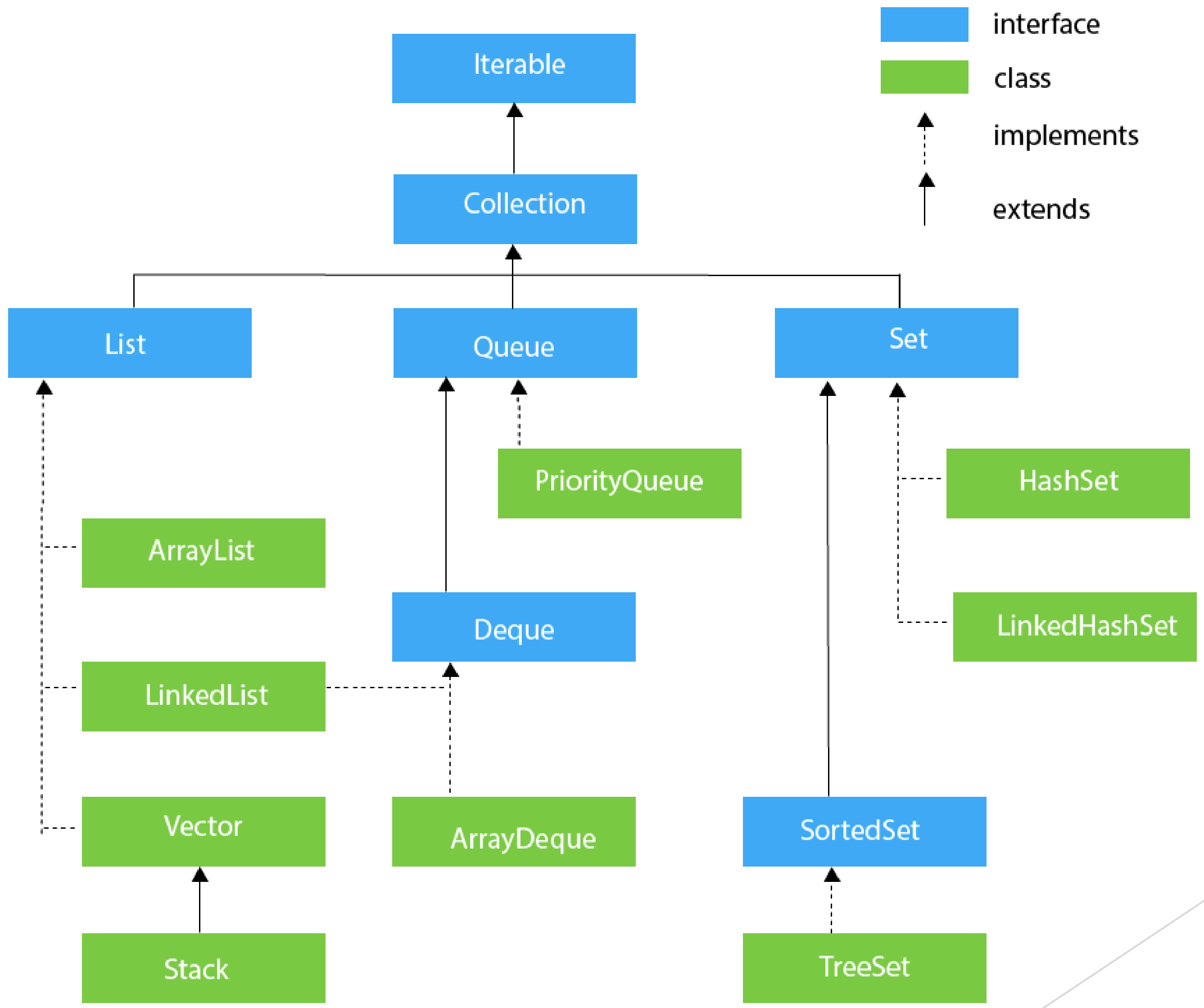
Java Collections

- The Collection in Java is a framework that provides an architecture to store and manipulate the group of objects
- Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.
- Java Collection means a single unit of objects.
- Java Collection framework provides many interfaces (**Set, List, Queue, Deque**) and classes (**ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet**).

What is a framework in Java

- It provides readymade architecture.
- It represents a set of classes and interfaces.
- It is optional.

Hierarchy of Collection Framework



Methods of Collection interface

No.	Method	Description
1	public boolean add(E e)	It is used to insert an element in this collection.
2	public boolean addAll(Collection<? extends E> c)	It is used to insert the specified collection elements in the invoking collection.
3	public boolean remove(Object element)	It is used to delete an element from the collection.
4	public boolean removeAll(Collection<?> c)	It is used to delete all the elements of the specified collection from the invoking collection.
5	default boolean removeIf(Predicate<? super E> filter)	It is used to delete all the elements of the collection that satisfy the specified predicate.
6	public boolean retainAll(Collection<?> c)	It is used to delete all the elements of invoking collection except the specified collection.
7	public int size()	It returns the total number of elements in the collection.
8	public void clear()	It removes the total number of elements from the collection.

Methods of Collection interface (contd...)

9	public boolean contains(Object element)	It is used to search an element.
10	public boolean containsAll(Collection<?> c)	It is used to search the specified collection in the collection.
11	public Iterator iterator()	It returns an iterator.
12	public Object[] toArray()	It converts collection into array.
13	public <T> T[] toArray(T[] a)	It converts collection into array. Here, the runtime type of the returned array is that of the specified array.
14	public boolean isEmpty()	It checks if collection is empty.
15	default Stream<E> parallelStream()	It returns a possibly parallel Stream with the collection as its source.
16	default Stream<E> stream()	It returns a sequential Stream with the collection as its source.
17	default Spliterator<E> spliterator()	It generates a Spliterator over the specified elements in the collection.
18	public boolean equals(Object element)	It matches two collections.
19	public int hashCode()	It returns the hash code number of the collection.

Homework: In your Laptop

Installing Java:

- Go to the Oracle Java website: https://www.oracle.com/java/technologies/javase-downloads.html
- Click on the "Download" button under the "JDK" column for the version of Java you want to install.
- Accept the license agreement and choose the appropriate operating system version for your computer.
- Once the download is complete, run the installer and follow the instructions to install Java on your computer.

Installing JetBrains IntelliJ IDEA:

- Go to the JetBrains IntelliJ IDEA website: https://www.jetbrains.com/idea/download/
- Click on the "Download" button for the community edition (which is free) or the ultimate edition (which has a free trial).
- Choose the appropriate operating system version for your computer.
- Once the download is complete, run the installer and follow the instructions to install IntelliJ IDEA on your computer.

After you have installed Java and IntelliJ IDEA

- Open IntelliJ IDEA.
- Click "Create New Project".
- Select "Java" and click "Next".
- Choose a name and location for your project and click "Next".
- Select the Java SDK you installed earlier and click "Next".
- Choose the project type (e.g. "Console Application") and click "Next".
- Click "Finish" to create the new project.

Thank You