Code Java Extended

You Already Know

Course(s):

Core Java

Recap

- Explain the fundamentals of Java language
  - Classes and Objects
  - Methods and Variables

- Explain data types and their declarations
  - Primitive data types

- Use of conditional statements, loops, break statements, and continue statements
  - If, if-else, and if-else-if
  - While, for, and do-while

Recap

- Explain classes and objects
  - Object life cycle
  - Operators and their precedence

- Explain the use of core keywords
  - *static*
  - *final*
  - *this*

A Day in the Life of a Test Engineer

Due to company's frequent change in goals and based on Joe's skills, he is added to another sprint. This sprint focuses majorly on implementing the features to their website. Selling the products is one of the decisions the stakeholders agreed to implement. In this sprint, Joe is assigned a task on JIRA which is a project management tool.

Joe has to develop a feature where users can add multiple items to the cart and delete items from the cart. However, a senior developer will be assisting Joe to complete this task. During pair programming, Joe was asked to write a program that does the basic operations similar to a calculator to get started.

In this lesson, we will learn how to solve this real-world scenario to help Joe complete his task effectively and quickly.

Learning Objectives

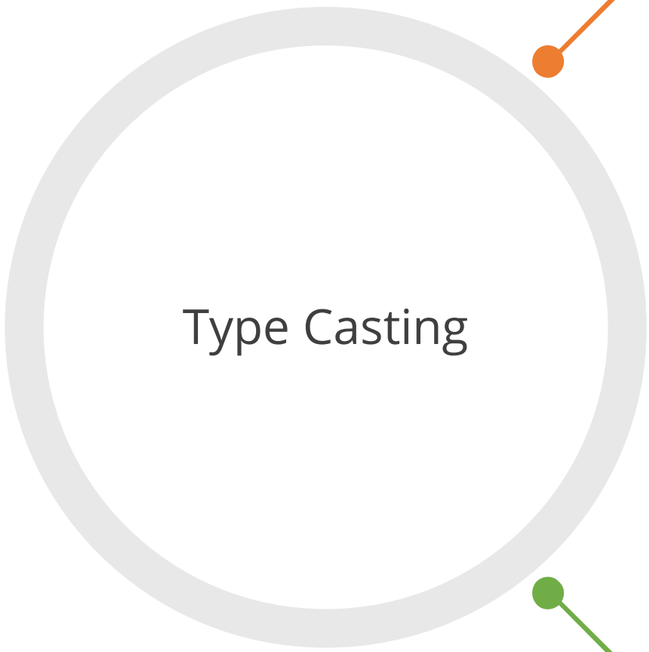
By the end of this lesson, you will be able to:

- Demonstrate type casting
- Use access modifiers
- Demonstrate working of while loop, do while loop, and for loop
- Explain object-oriented concepts of Java
- Describe how exceptions are handled using try-catch block, finally block, throw keyword, and throws keyword

Type Casting

Type Casting

When two data types are compatible with each other, the value of one data type that is assigned to the other is called type casting.



Implicit Type Casting:

byte → short → int → long → float → double

Explicit Type Casting:

double → float → long → int → short → byte

Type Casting: Types

Widening or Implicit Conversion

- Widening or implicit conversion takes place when two data types are automatically converted
- This is possible when:
  - The two data types are compatible
  - The value of a smaller data type is assigned to a bigger data type
- Conversion of numeric data type to char or boolean is not supported

Narrowing or Explicit Conversion

- To assign the value of a large data type to a smaller data type, we have to perform narrowing or explicit conversion
- This is useful for incompatible data types when automatic conversion is not possible

Type Casting

Duration: 20 min.

Problem Statement:

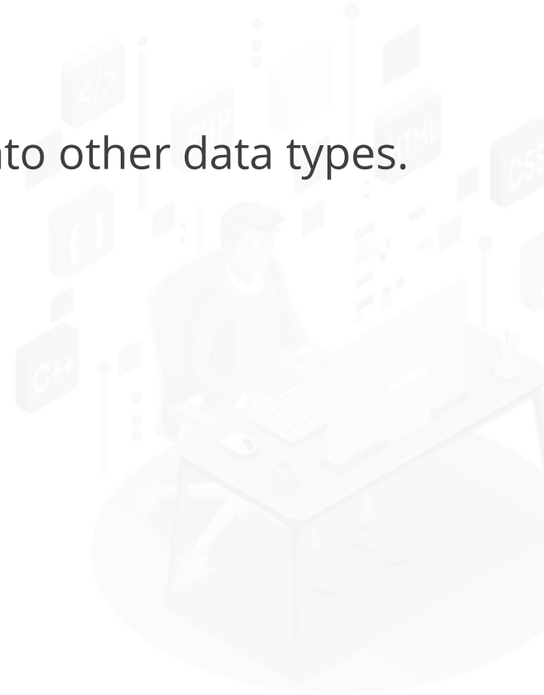
Write a program which will take a string as input and will convert it into other primitive data types.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to demonstrate type casting:

1. Create a Java project in your IDE.
2. Write a program in Java to take an input from user and convert the input variable into other data types.
3. Initialize the .git file.
4. Add and commit the program files.
5. Push to code to your GitHub repository.



Access Modifiers

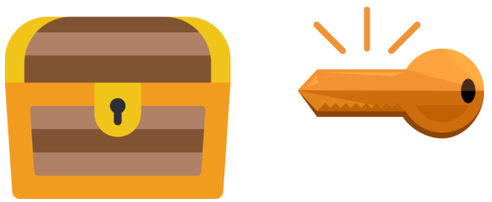
Access Modifiers

The access modifiers specify the accessibility of a data member, class, constructor, or method.

Modifier	Class	Package	Subclass	World
public	✓	✓	✓	✓
protected	✓	✓	✓	✗
No modifier	✓	✓	✗	✗
private	✓	✗	✗	✗

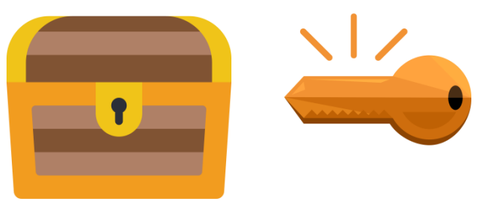
Types of Access Modifiers

Default



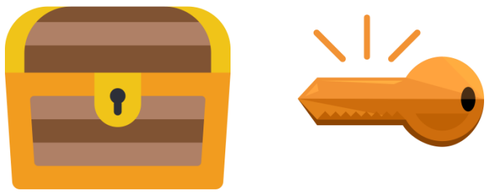
The default access modifiers are accessible only within the package.

Public



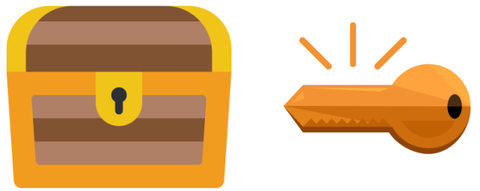
It is accessible everywhere. It has the widest scope.

Private



Private access modifiers are accessible only within the class.

Protected



The protected access modifiers are accessible only within and outside the package but only through inheritance.

Access Modifiers

Duration: 20 min.

Problem Statement:

Write a program to demonstrate how and when access modifiers are used.

ASSISTED PRACTICE

Assisted Practice: Guidelines

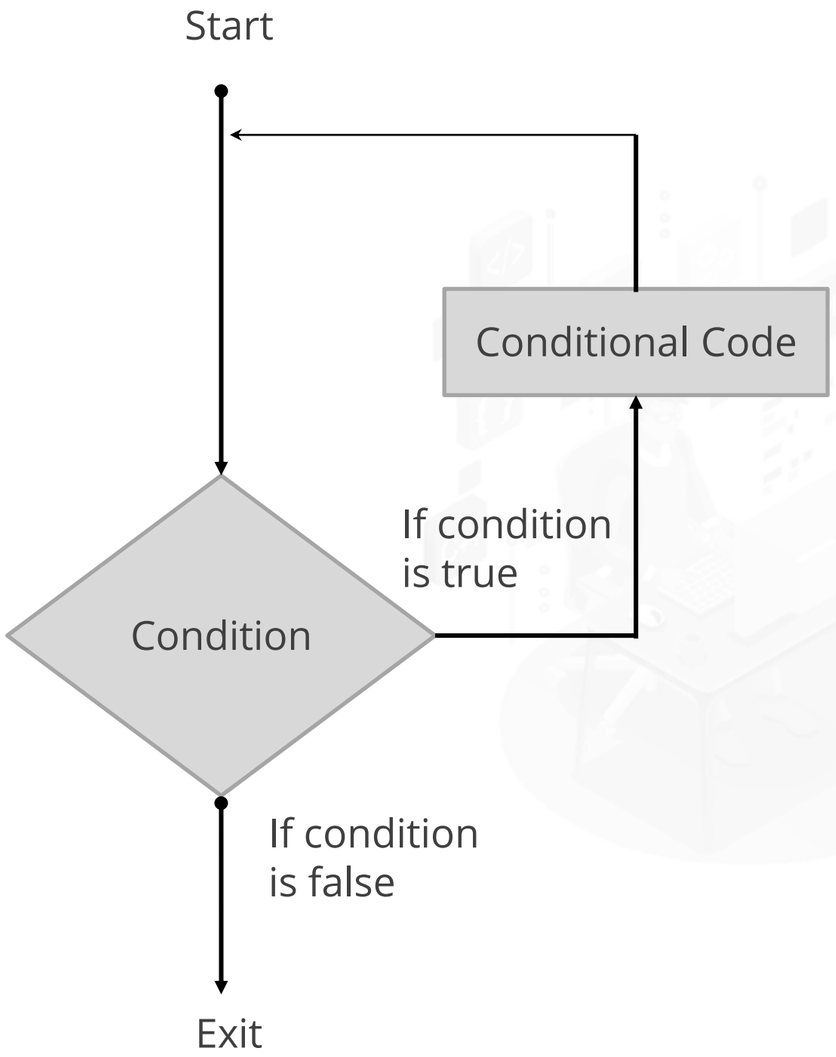
Steps to demonstrate access modifiers:

1. Create a Java project in your IDE.
2. Write a program in Java to create variables with different access modifiers and access these variables in main().
3. Initialize the .git file.
4. Add and commit the program files.
5. Push to code to your GitHub repository.

Looping

Loops in Java

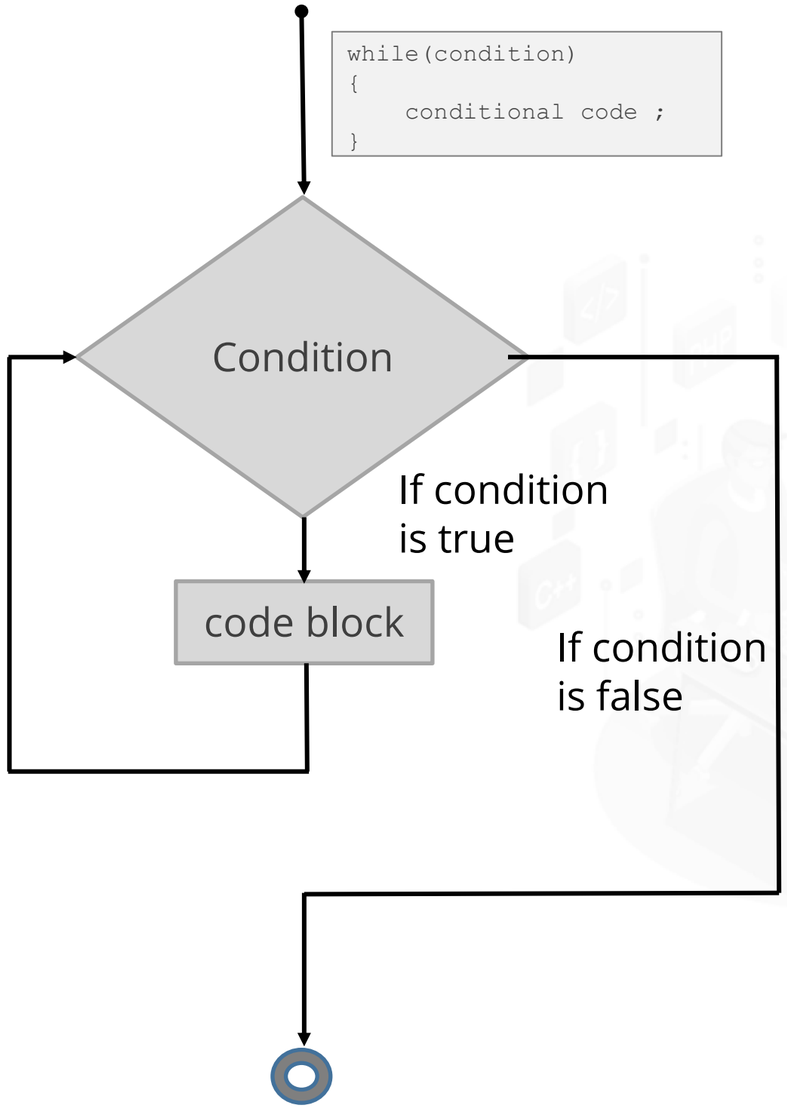
- Enable you to execute blocks of statements multiple times
- Controlled by Boolean expressions
- Types of loops:
  - While loop
  - Do while loop
  - For loop



while Loop

- Continually executes a block of statements as long as a given statement is true
- Evaluates its expression at the top of the loop
- Has the following syntax:

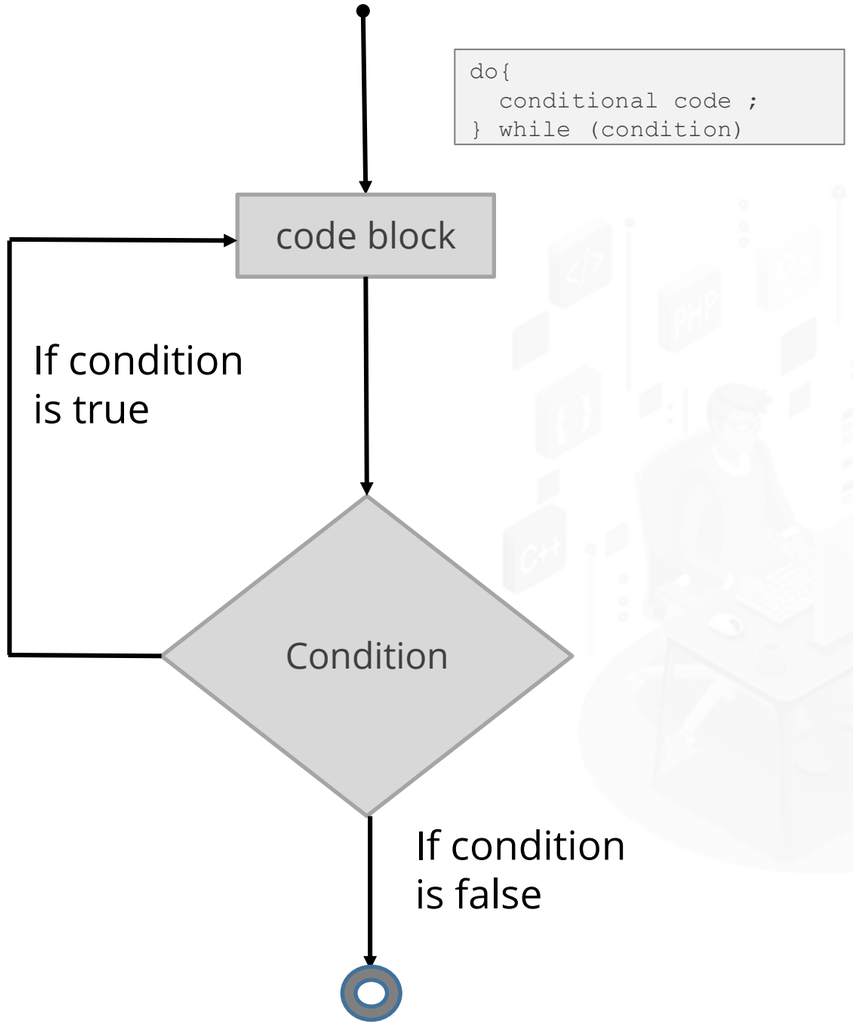
```
while (expression) {
    statement(s)
}
```



do...while Loop

- Same as **while** loop, except that it is guaranteed to execute at least one time
- Evaluates its expression at the bottom of the loop, instead of the top
- Has the following syntax:

```
do {
    statement(s)
} while (expression);
```



for Loop

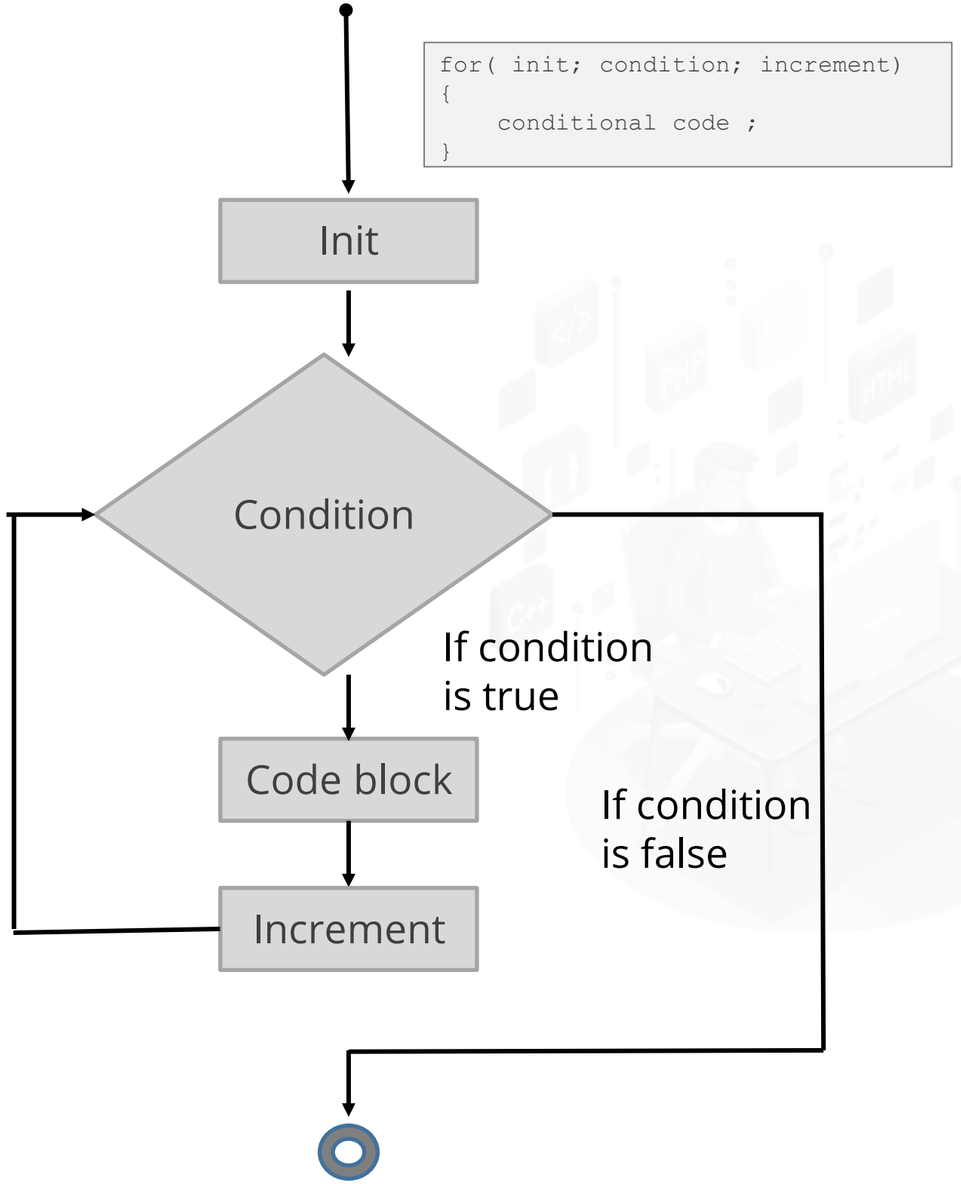
- Commonly used for simple iterations
- Allows you to repeat a set of statements a certain number of times until a condition is matched
- Has the following syntax:

Initialization is executed once the loop begins

Increment is invoked after each iteration through the loop

```
for (initialization; termination; increment) {
    statement(s)
}
```

If the condition is false, then the for loop will be terminated



While Loop

Duration: 15 min.

Problem Statement:

Write a program to demonstrate **while** loop execution.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to demonstrate **while** loop:

1. Create a Java project in your IDE.
2. Write a program in Java to demonstrate while loop execution.
3. Initialize the .git file.
4. Add and commit the program files.
5. Push to code to your GitHub repository.

Do While Loop

Duration: 15 min.

Problem Statement:

Write a program to demonstrate **do while** loop execution.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to demonstrate **do while** loop:

1. Create a Java project in your IDE.
2. Write a program in Java to demonstrate do while loop execution.
3. Initialize the .git file.
4. Add and commit the program files.
5. Push to code to your GitHub repository.

For Loop

Duration: 20 min.

Problem Statement:

Write a program to demonstrate **for** loop execution.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to demonstrate **for** loop:

1. Create a Java project in your IDE.
2. Write a program in Java to demonstrate for loop execution.
3. Add and commit the program files.
4. Push to code to your GitHub repository.

Class, Objects, and Constructors

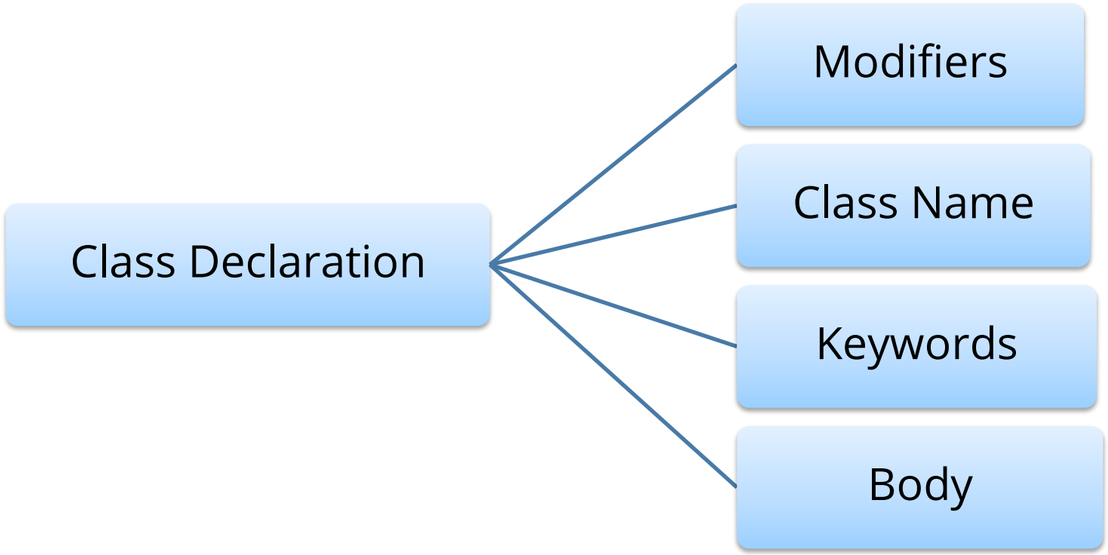
What Is a Class?

A class is a user defined template or prototype that defines objects of the same type. All objects in the class will exhibit the same properties.

Class	Object	Common Class Properties
Tree	Oak	Branches Leaves Roots
	Willow	
	Elm	

What Is a Class?

A class declaration consists of the following segments:



Class Initialization

Class initialization consists of a block of code preceded by the “static” keyword. Every time a class is loaded, Java invokes this code.

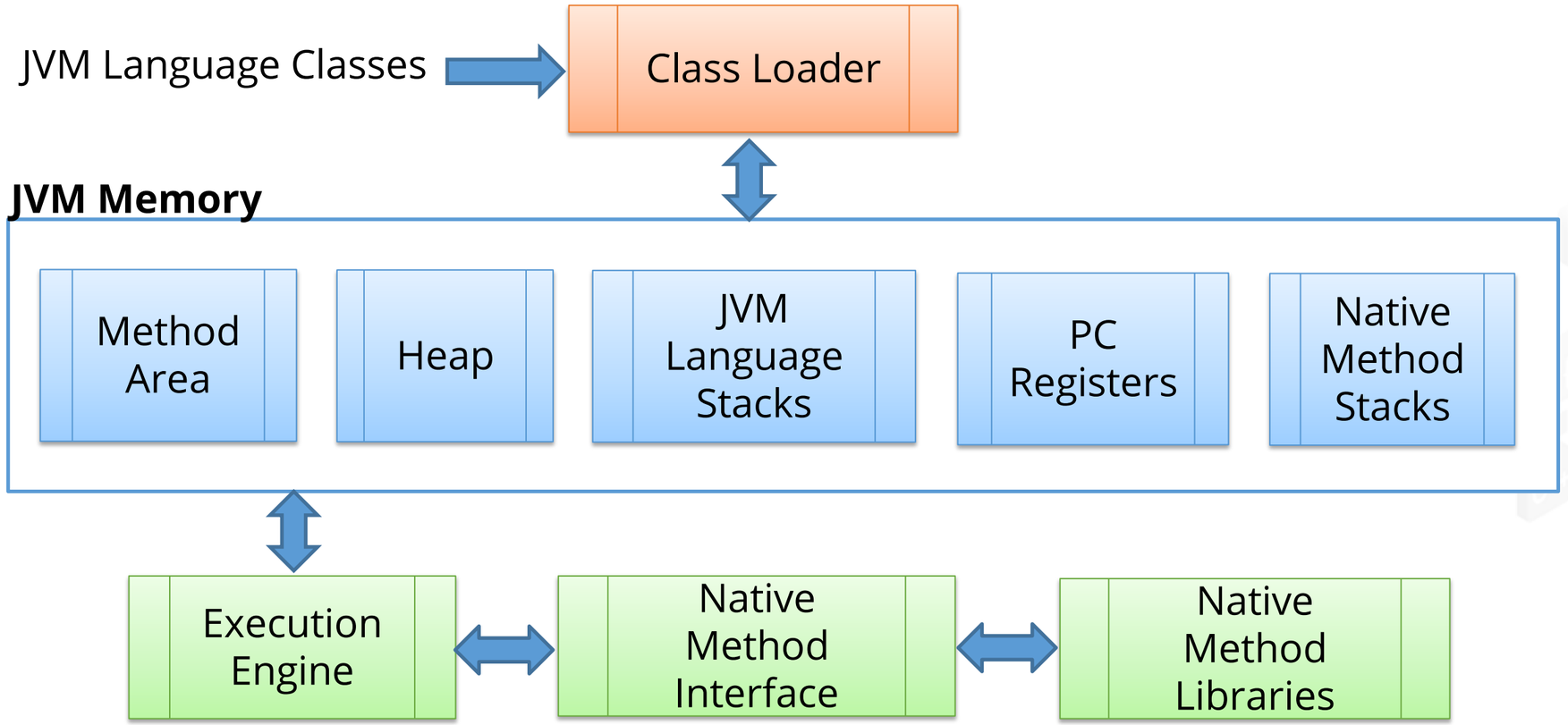
```
class Demo {
      static {
        // ...initialization code here...
      }
 }
```

What is class instantiation?

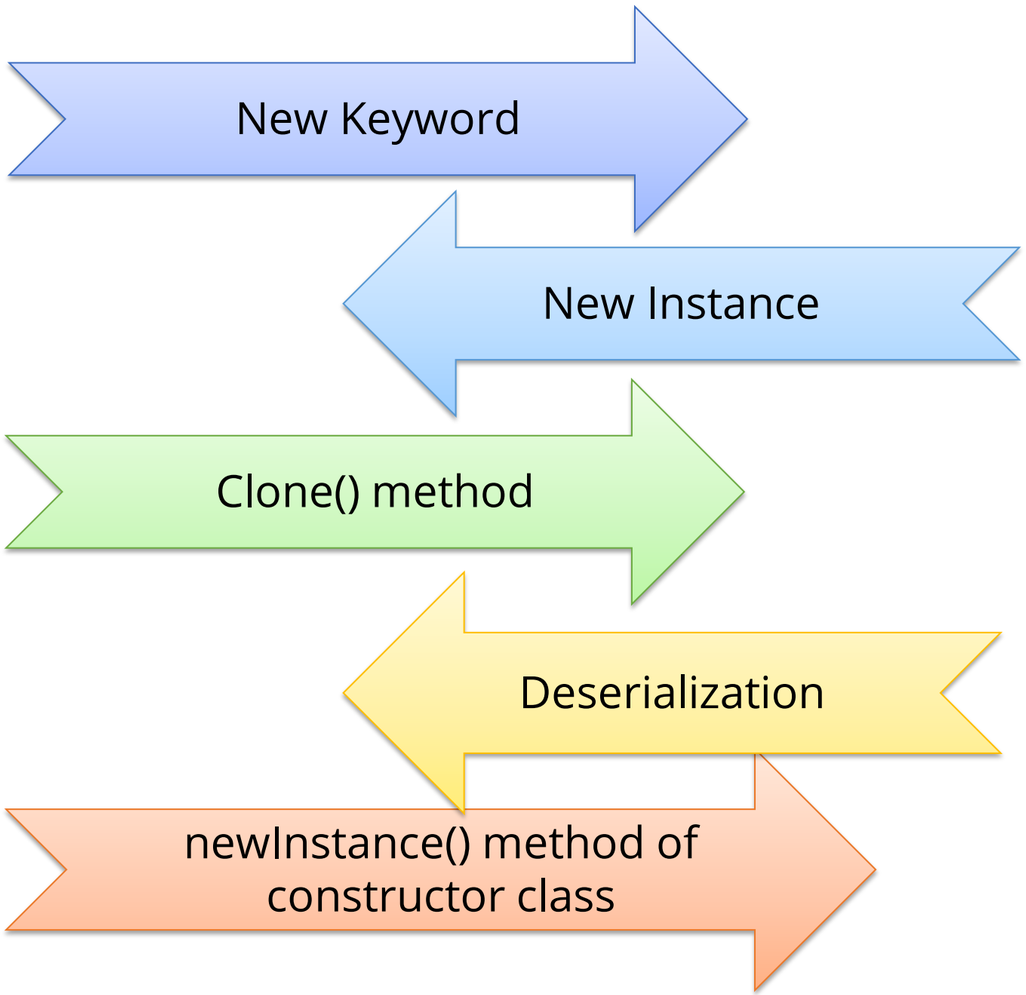
Creating a new instance (object) in a class is called class instantiation.

```
class Demo {
       {
        // ...initialization code here...
       }
 }
```

JVM Architecture



Ways to Instantiate a Class



Difference between Class Referencing and Object Creation

- An object is an instance created in a class and a reference is the memory address to navigate to that said object.

- For example: A mobile phone has several contacts, each contact representing an actual person.
- **Object**: Person

- **Reference**: contact, which is a means of reaching the person.

```
class Person {
    String name = "Smith";
    int num = 9990000999;
      public static void main(String[] args)
      {
      Person p = new Person();
      }
}
```

p is a reference which points out the object to the person.

Static and Instance Blocks

Static Blocks:

- Used for static initializations of a class
- Executed when a static member is invoked in a class
- Executed before constructors

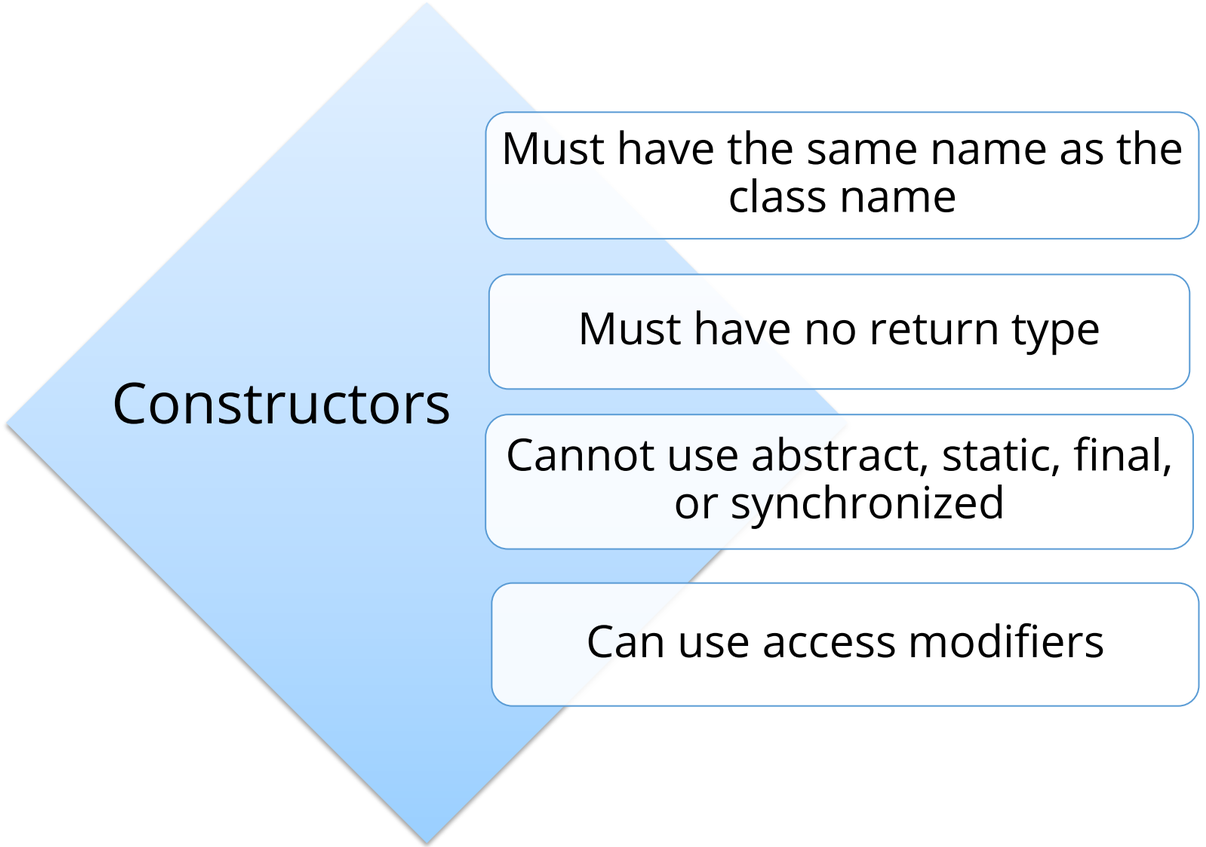
Instance Blocks:

- Executed when an object is created in a class

```
class student
{
      static int rollno ;
      static String name;
      static{ rollno = 28; name = "John";
      }
      public static void main(String args[]){}
}
```

What Are Constructors?

Constructors are blocks of code executed while initializing an object. Constructors are used to assign value to class variables when creating an object.

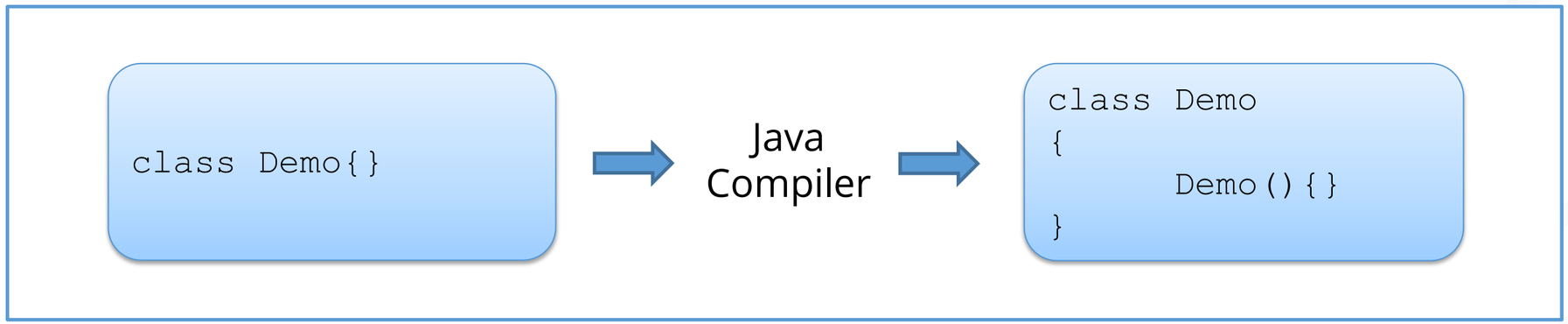


Types of Constructors

There are three types of constructors such as Default, No-Arg, and Parameterized.

Default Constructor:

If user does not implement any constructor, then default constructor is inserted in the class by Java compiler.



Types of Constructors

No-Arg Constructor:

These constructors have no arguments.

```
class Demo
{
      public Demo(){}
      public static void main(String arg[]){
      new Demo();
      }
}
```

Types of Constructors

Parameterized Constructor:

These constructors have arguments or parameters.

```
public class Employee
{
	int empId;
	String empName; //parameterized constructor with two parameters
	Employee(int id, String name){
		this.empId = id;
		this.empName = name;
		}
}
```

Class, Objects, and Constructors

Duration: 20 min.

Problem Statement:

Explain Classes, Objects, and Constructors using an example.

ASSISTED PRACTICE

Assisted Practice: Guidelines

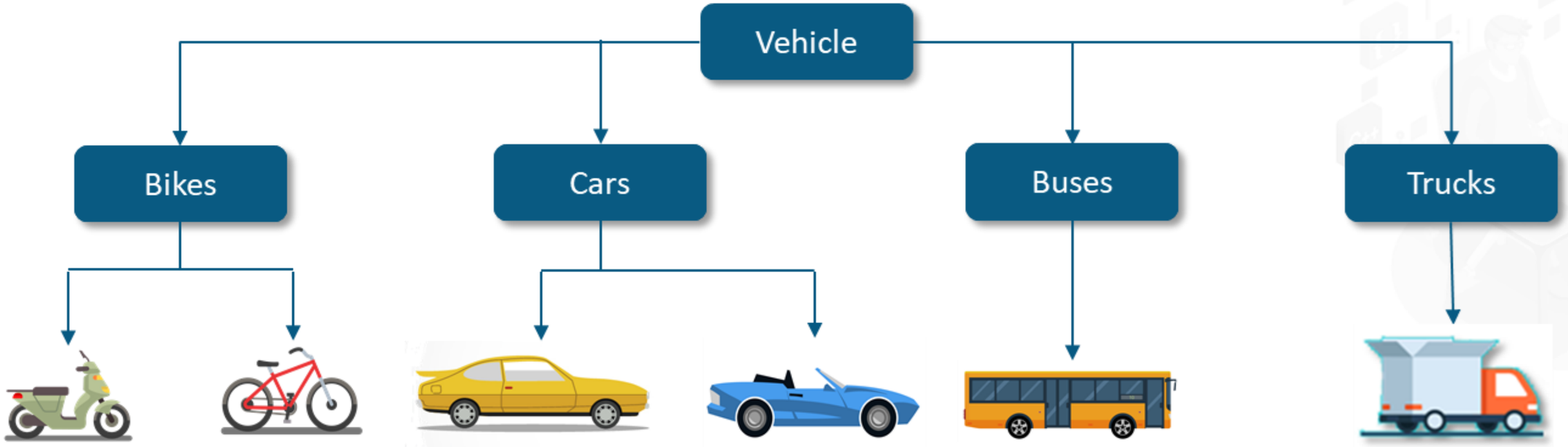
Steps to demonstrate class, objects, and constructors:

1. Create a Java project in your IDE.
2. Write a program in Java to create a class and its objects in main().
3. Initialize the .git file.
4. Add and commit the program files.
5. Push to code to your GitHub repository.

Inheritance

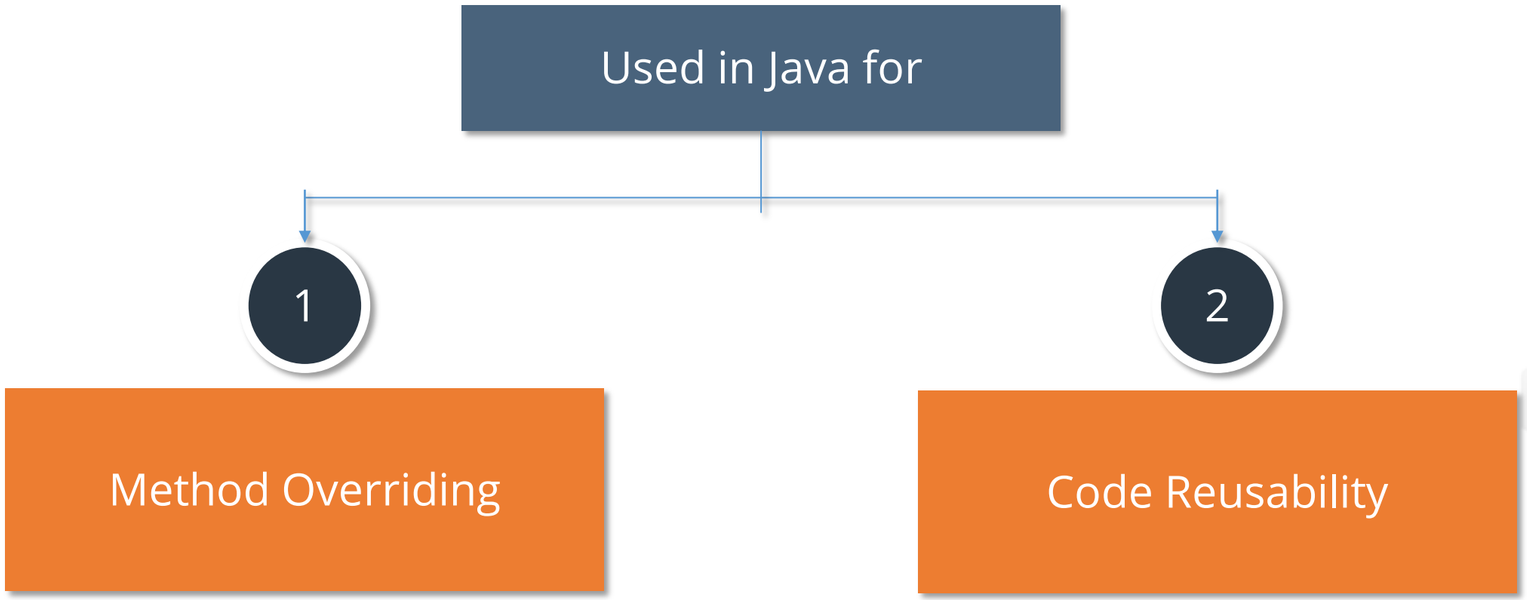
Inheritance

- A mechanism in which one class acquires all the properties and behaviors of the parent class
- Used for method overriding and code reusability



IS-A Relationship

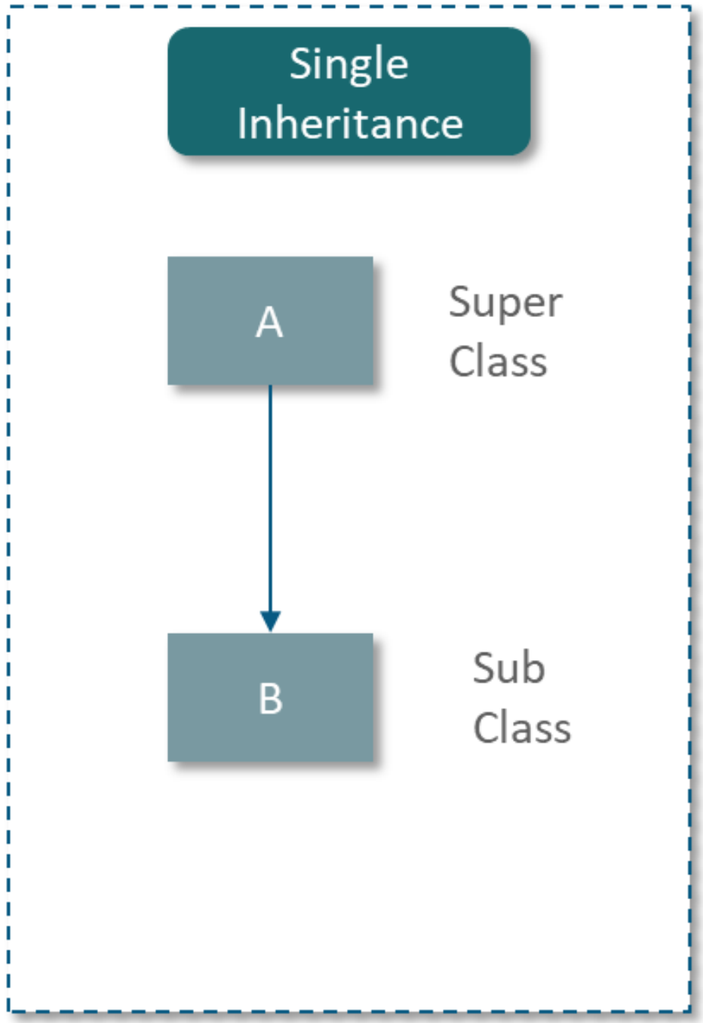
Inheritance represents IS-A relationship, also known as Parent-Child relationship.

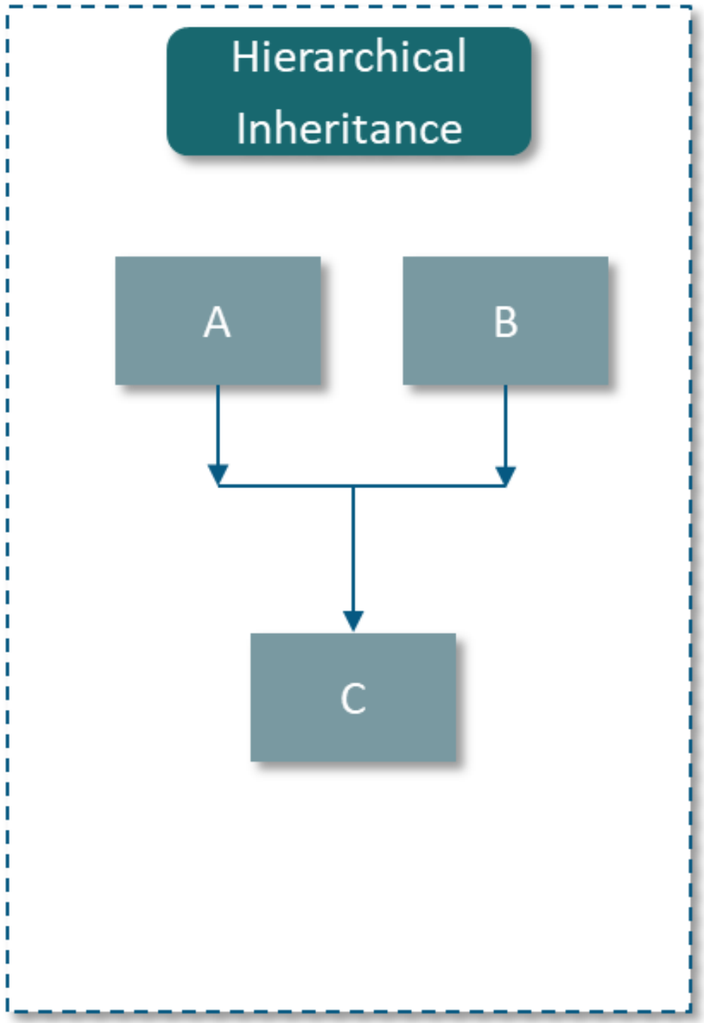


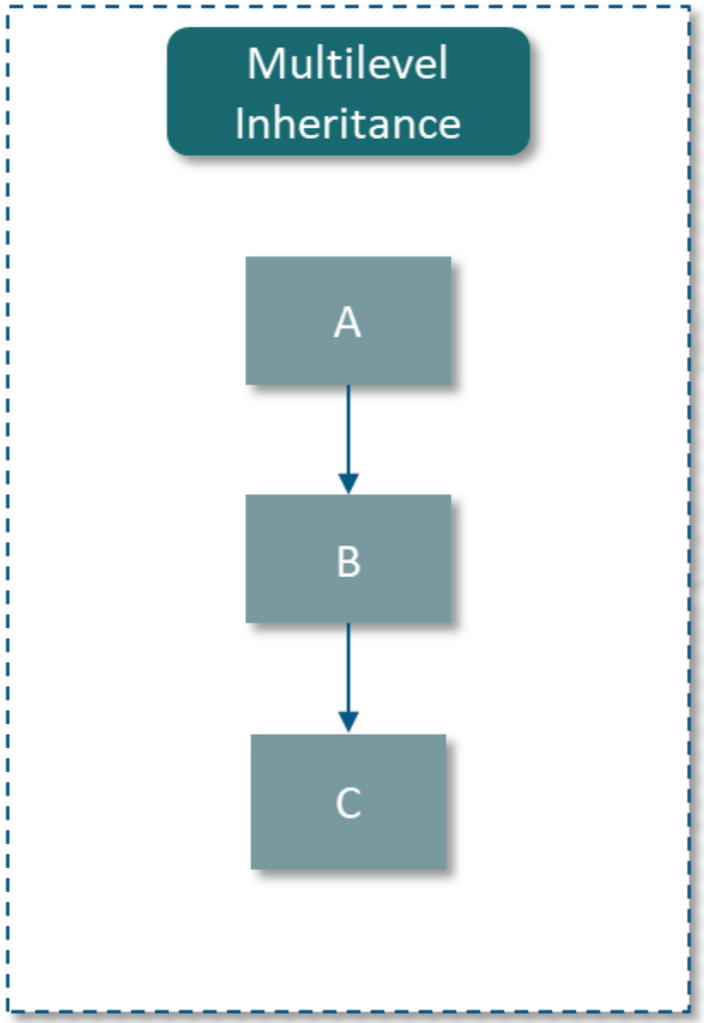
Syntax of Inheritance

```
Class SubClass extends SuperClass
{
        //Variables
        //Methods
}
```

Types of Inheritance







Inheritance

Duration: 15 min.

Problem Statement:

Explain types of inheritance using examples.

ASSISTED PRACTICE

Assisted Practice: Guidelines

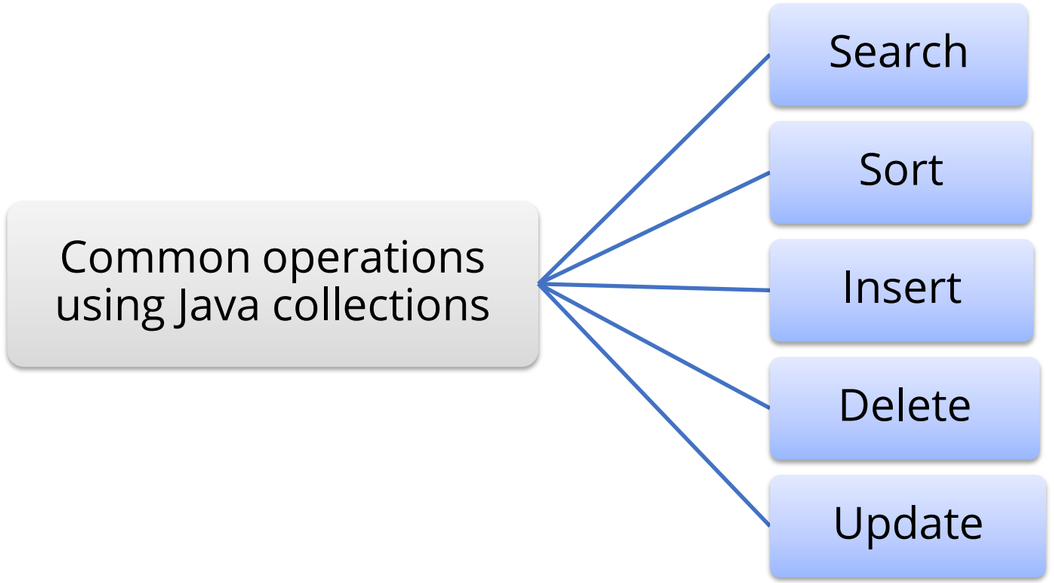
Steps to demonstrate inheritance:

1. Create a Java project in your IDE.
2. Write a program in Java to demonstrate inheritance.
3. Initialize the .git file.
4. Add and commit the program files.
5. Push to code to your GitHub repository.

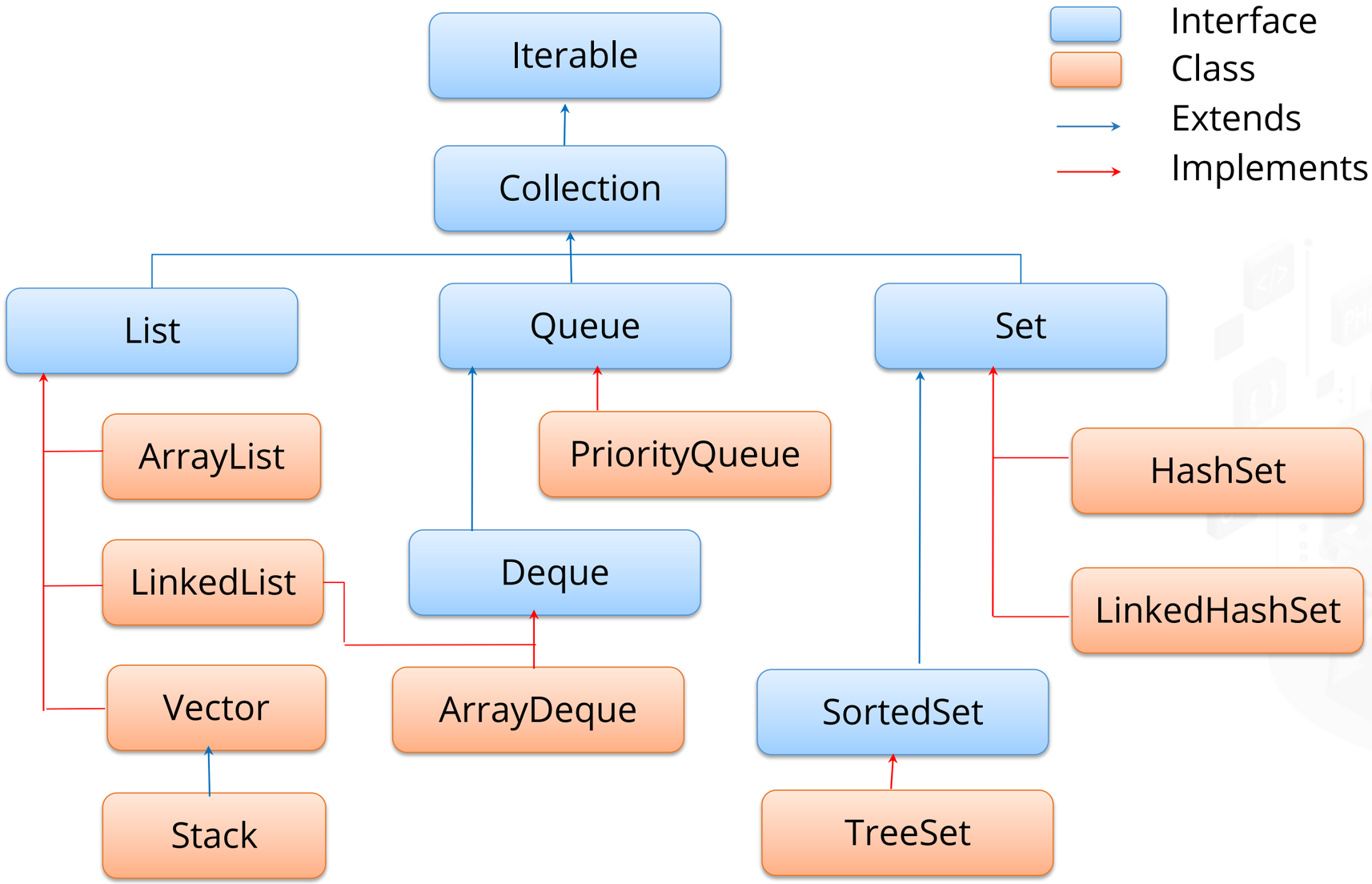
Collections and Generics

What Are Collections?

A collection is a group or single unit of individual objects. Java Collection Framework defines several classes and interfaces to represent objects as a single unit.



Collection Hierarchy



Legacy Classes vs. Collection Framework

Legacy Classes	Collection Framework
Legacy classes comprises classes and interfaces supported by older versions of Java.	Collections are used to store and manipulate group of objects.
Legacy classes are: • Dictionary • Hashtable • Properties • Stack • Vector	Comes with interfaces like Collection, Set, List, or Map. Classes include ArrayList, LinkedList, Vector and Stack.
Legacy classes are synchronized.	Collections are non-synchronized.

List, Set, LinkedList, and Queue

List

- An interface that contains sequentially arranged elements
- ArrayList, LinkedList, and Vector subclasses implement List interface

Set

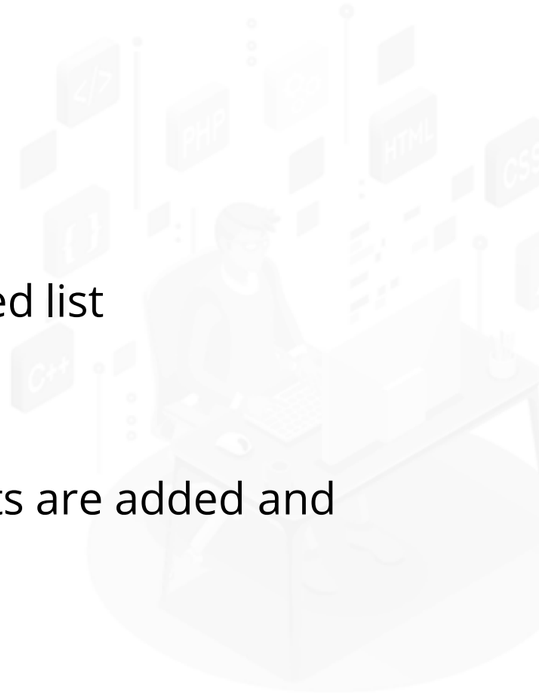
- Contains unique elements arranged in any order
- HashSet, LinkedHashSet, TreeSet classes implement Set interface

LinkedList

- Contains elements of a specified collection, arranged into an ordered list

Queue

- Contains an ordered list of homogeneous elements, where elements are added and removed at the rear and front end respectively
- LinkedList, Priority queue, ArrayQueue implement this interface



Collections Methods

Here are some collection methods:

- size()
- add(Object o)
- addAll(Collection c)
- remove(Object o)
- removeAll(Collection c)
- contains(Object o)
- containsAll(Collection c)
- clear()
- isEmpty()
- iterator()
- equals(Object o)
- toArray()
- toArray(Object[] a)
- retainAll(Collection c)

What Are Generics?

Generics ensure type safety in Java by making bugs detectable during compile time. Type Erasure feature in Java Generics erases extra information added to source code.

Types of Generics:

Generic type class or interface: A class is generic if it declares one or more type variables

Generic type method or Constructor: Generic methods have their own parameters

Applications

The Java collection hierarchy has a number of uses, such as:

- ArrayList can be used for finding a specific object using array index.
- Linked list is useful in insertion and deletion operations.
- Set can be used to create unique elements and TreeSet is an implementation of the set.

Collections and Generics

Duration: 30 min.

Problem Statement:

Write a program to demonstrate working of Collections and Generics.

ASSISTED PRACTICE

Assisted Practice: Guidelines

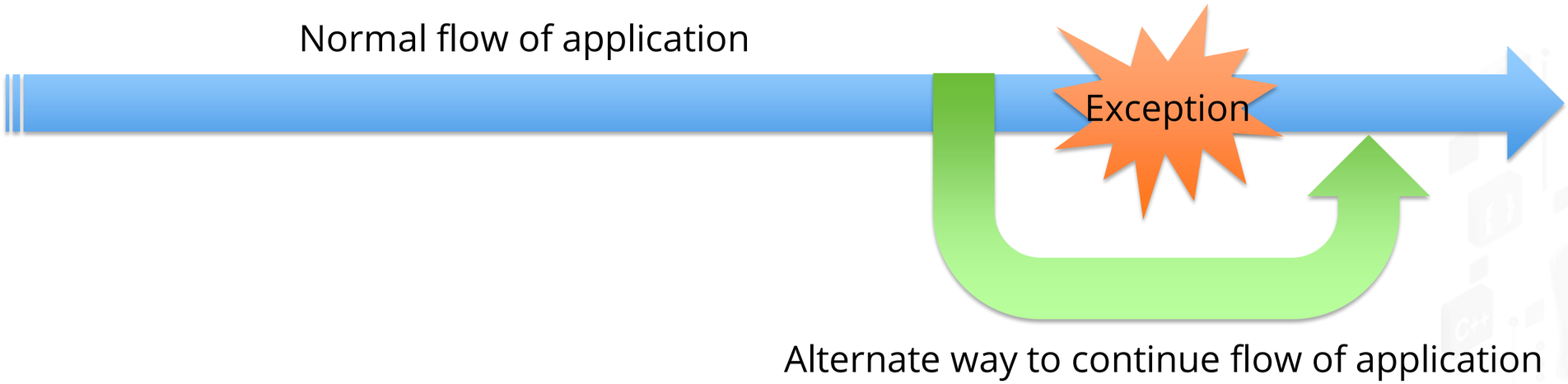
Steps to demonstrate Collections and Generics:

1. Create a Java project in your IDE.
2. Write a program to demonstrate functionality of Collections.
3. Initialize the .git file.
4. Add and commit the program files.
5. Push to code to your GitHub repository.

Exception Handling

What Is Exception Handling?

Exception Handling is a mechanism to handle run time errors that disrupt the normal flow of an application.

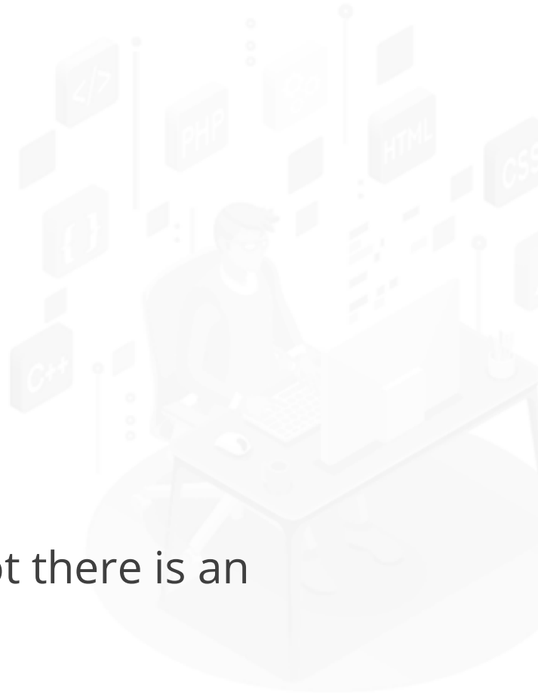


Several factors can cause an exception during runtime, such as an invalid input data, a nonexistent file, or an interrupted network connection.

Keywords Used for Exception Handling

Following keywords are used for exception handling:

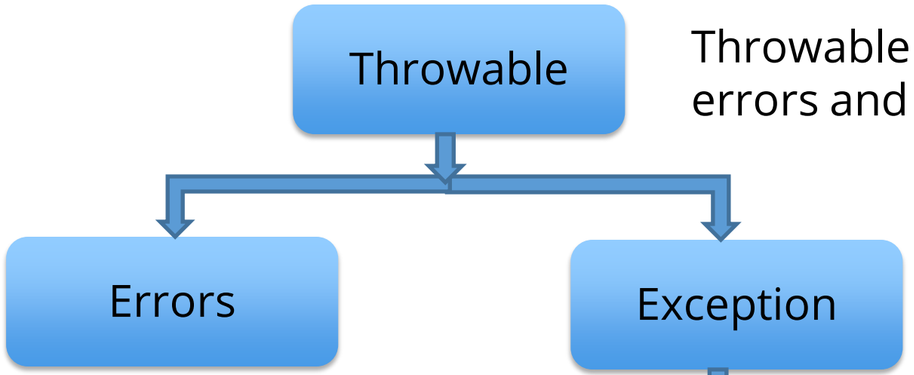
try	A block that contains a set of statements where an exception may occur
catch	Handles the uncertain exception in the **try** block
throw	A keyword manually transfers control from **try** to **catch**
throws	A keyword handles exceptions without **try** and **catch**
finally	block contains a set of statements that will always execute whether or not there is an exception



Need of Handling Errors and Exception

Why errors and exceptions should be handled?	When an exception occurs midway through execution, the flow of the statement breaks and the program terminates abruptly. Exception and error handling ensures that all statements run and thereby maintains the flow of the program.
	The traditional error handling methods use multiple *if else* conditions in the normal program flow, which makes the code less readable. Exception handling separates error handling code from the regular code, using *try catch* blocks.

Exception Hierarchy



Throwable is the superclass of all errors and exceptions.

Errors are serious issues that cannot be handled. Errors result in the abrupt termination of the program.

Checked exceptions can be foreseen during compile time and can be handled before run time.

Unchecked exceptions occur during runtime and need to be handled by the developer.

Types of Exception

ArithmeticException	Thrown when an exception has occurred in an arithmetic operation
ArrayIndexOutOfBoundsException	Thrown to notify that an array has been accessed with an illegal index
ClassNotFoundException	Thrown when a class whose definition is not found, is accessed
FileNotFoundException	Thrown when a file is not found or is inaccessible
NullPointerException	Thrown when accessing members of a null object
RuntimeException	Represents any exception that occurs during runtime
IOException	Thrown when an input-output operation fails or is interrupted
NumberFormatException	Thrown when the conversion from string to numeric format fails
NoSuchFieldException	Thrown when the class does not contain the specified field or variable

Basic Try-Catch Block

Duration: 15 min.

Problem Statement:

Demonstrate how a basic **try-catch** block is used for Exception Handling.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to demonstrate basic **try-catch** block:

1. Create a Java project in your IDE.
2. Write a program to demonstrate functionality of try-catch block.
3. Initialize the .git file.
4. Add and commit the program files.
5. Push to code to your GitHub repository.

Throw and Throws

Duration: 20 min.

Problem Statement:

Explain **throw** and **throws** using an example.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to demonstrate **throw** and **throws** keywords:

1. Create a Java project in your IDE.
2. Write a program to demonstrate functionality of throw and throws keywords.
3. Initialize the .git file.
4. Add and commit the program files.
5. Push to code to your GitHub repository.

Try with Parameters

Duration: 30 min.

Problem Statement:

Implement **try** with parameters in a program.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to demonstrate **try** with parameters:

1. Create a Java project in your IDE.
2. Write a program to demonstrate working of try block.
3. Initialize the .git file.
4. Add and commit the program files.
5. Push to code to your GitHub repository.

Multiple Catch with Exception Class Hierarchy

Duration: 30 min.

Problem Statement:

Implement multiple **catch** with **exception class** hierarchy in a program.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to demonstrate multiple **catch** with **exception class** hierarchy:

1. Create a Java project in your IDE.
2. Write a program to demonstrate working of multiple catch blocks..
3. Initialize the .git file.
4. Add and commit the program files.
5. Push to code to your GitHub repository.

Finally Block

Duration: 30 min.

Problem Statement:

Demonstrate the use of **finally** block in a program.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to demonstrate **finally** block:

1. Create a Java project in your IDE.
2. Write a program to demonstrate working of *finally* block.
3. Initialize the .git file.
4. Add and commit the program files.
5. Push to code to your GitHub repository.

Key Takeaways

- Java is a general-purpose, object-oriented, concurrent, and the run time environment (JRE) programming language
- Java can be used in web development, Big Data, continuous testing, and frameworks
- Type casting is used to assign value of one data type to the other, only if the two data types are compatible
- The access modifiers in java are used to specify the scope of a field, method, constructor, or a class
- We can change the scope of the field, method, constructor, or a class by applying the access modifiers on it

Create a program to find areas of Multiple Dimensions

Problem Statement:

Duration: 90 min.

You are asked to develop a functionality where user will enter the dimensions and you need to calculate and print the area of selected shape.

LESSON-END PROJECT

Before the Next Class

Course: SQL Training

You should be able to:

- Explain what is database
- Demonstrate SQL queries
- Filter results using queries
- Group data using queries