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

Types of Type Casting

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.

Types of Access Modifiers

Default

The default access modifiers are accessible only within the package.

Public

It has the widest scope and is accessible everywhere.

Private

Private access modifiers are accessible only within the class.

Protected

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

Methods

A method is a collection of statements, which performs specific tasks and returns results to the caller. It allows you to write reusable code and divide the program into several small units.

Return statement

• The return statement is a control flow statement, which terminates the execution of method and returns control to its caller. • If a method has a void return type, it means the method does not return anything.

Calling a Method

Call by Value

In the Call by Value method of passing arguments to a function, the program copies the actual value of an argument into the formal parameter of a function.

Method Overloading

A class may define multiple methods with the same name and return type, but different number of arguments or arguments of different data types.

Constructors

Constructors are used to initialize an object when it is created. It has the same name as the class but no explicit return type.

No Argument Constructor:

• It does not have any parameter. • It is also called “Default” constructor. • If constructor is not defined in a class, then compiler creates a default constructor (with no arguments) for the class

Parameterized Constructor:

• It can have multiple parameters. • To initialize fields of the class with own values, use parameterized constructor

Difference between Constructor and Method

Constructor and method

Constructor must not have the return type. Method must have the return type.

Constructor name must be same as the class name. Method name must not be same as the class name. Constructor is used to initialize the state of an object. Method is used to expose the behavior of an object.

Constructor is invoked implicitly. Method is not invoked implicitly.

Collections

A Java collection framework provides an architecture to store and manipulate a group of objects

List

A list is an ordered collection of elements which may contain duplicates. Lists are further classified into the following: LinkedList, ArrayList, Vector

Queue

Queue is a data structure which follows first in first out (FIFO) algorithm.

A PriorityQueue class allows you to initialize a queue: Queue B = new PriorityQueue();

Another way to initialize queue : Queue A = new LinkedList();

Set

• A set refers to a collection that cannot contain duplicate elements. • It is mainly used to model the mathematical set abstraction. • Set has its implementation in various classes such as HashSet, TreeSet, and Linked HashSet.

Map

• The map interface maps unique keys to values. • It contains a key that is an object used to retrieve a value. • A map object only stores key-value pairs.

Inner Classes

An inner class is a class which is declared inside another class.

Java inner class can be declared as public, private, and protected.

Java inner classes are nothing but non-static nested classes.

Java inner classes are used to achieve security mechanism.

An inner class can access all the data members and member functions of outer class including private data members.

Less code is required to write an inner class.

Strings

Strings are objects that represent the sequence of characters. It cannot be changed once created. It is not a primitive data type.

Syntax String s=“NOAH” String s=new String(“NOAH”)

StringBuffer and StringBuilder

• StringBuffer is mutable. It means one can change the value of the object. • The objects created using StringBuffer are stored in the heap. • It is thread-safe. • Its performance is affected due to its thread-safe property. Hence, StringBuffer is slower than StringBuilder. StringBuffer sbr= new StringBuffer (“example1”);

• StringBuilder is mutable like StringBuffer. • It is not thread-safe. • StringBuilder is faster than StringBuffer. StringBuilder slr= new StringBuilder(“example2”);

Arrays

An array is a data structure used to store elements of the same data type. It is index-based. The first element refers to index 0. You can create one dimensional as well as multidimensional arrays in Java.

In single-dimensional, arrays elements are stored in rows only int a= new int[5];

In multidimensional arrays, elements are stored in the form of rows and columns. int s [ ] [ ]=new int [3] [3]

S[0][1] S[0][1] S[0][2]

S[1][0] S[1][1] S[1][2]

S[2][0] S[2][0] S[2][2]

Regular Expressions: A regular expression is an API used to define a pattern for searching or manipulating strings. Regular expression is widely used to define the constraint on strings like password and email validation.

Patterns

Patterns Description abc This sequence should be followed exactly [abc] Any one letter from a, b, or c should match [^abc] Any letter other than these three should match [a-z] Any letter from a to z can be present in the sequence [a-zA-z0-9] Any letter from a to z or A to Z or any digit from 0 to 9 can be in the sequence . Any character other than the line terminator can be in the sequence ^ To check if any character is present at the beginning of a line $ To check if any character is present at the end of a line \b To verify if any character is present at the word boundary or not \B To verify that no character is present at the word boundary \G To check that the character is present at the end of the previous match

Meta Characters

The meta-characters have predefined meanings and are used to make easier use of patterns

Patterns Description \d A digit [0-9] \D A non-digit[^0-9] \s A whitespace character \S A non-whitespace character \w A word character \W A non-word character[^\w]

Quantifiers

A quantifier defines how often an element can occur.

Patterns Description \* Occurs zero or more times + Occurs one or more times ? Occurs no or one time {X} Occurs X number of times {X,Y} Occurs between X and Y times \*? It tries to find the smallest match. This makes the regular expression stop at the first match

Concurrency and Multithreading

Concurrency is the ability to run multiple programs in parallel. It enables a program to achieve high performance and throughput by utilizing the untapped capabilities of the underlying operating system.

Multithreading feature allows concurrent execution of multiple parts of a program for maximum CPU utilization. Each part of such a program is called a thread

Mechanism to create threads

Extending the “Thread” class

Implementing the “Runnable” interface

Thread Safety

Thread safety is a process that makes the program safe to be used in a multithreaded environment. Different ways to implement thread safety are listed below: • Use of synchronization • Use of atomic wrapper classes from java.util.concurrent.atomic package • Use of locks from java.util.concurrent.locks package • Use of thread safe collection classes • Use of a volatile keyword with variables

The object class in Java has three final methods that allow threads to communicate about the locked status of a resource.

wait() It intimates the calling thread to halt the lock and sleep until some other thread enters the same monitor and calls.

notify() It wakes up a thread that called wait() on the same object. It intimates the waiting thread to wake up. It does not give up a lock on a resource.

notifyAll() It wakes up all the threads that called wait() on the same object. The highest priority thread executes in most of the situations provided.

Serialization and Deserialization

Serialization is a process of converting the state of an object into a byte stream. Deserialization is a reverse process of converting the byte stream to recreate the actual Java object in the memory.

Exception Handling

Asynchronous Exceptions

Deal with hardware and external problems • Mouse failure • Keyboard, motherboard failure • Memory problems • Power failure java.lang.Error is a super class of all asynchronous exceptions

Synchronous Exceptions

Deal with programmatic run-time errors Types: • Checked exceptions • Unchecked exceptions

Checked Exceptions

Arise from external conditions during compile time and are expected to be handled by the programmer Examples: “requested file not found” and “network failure”

Unchecked Exceptions

Arise from conditions that represent bugs during execution time or situations that are considered difficult to handle Example: “runtime exceptions are attempting to access beyond the end of an array”

Difference between Error Class and Exception Class

Errors Exceptions Errors in Java are of type java.lang.Error. Exceptions in Java are of type java.lang.Exception. All errors in Java are of unchecked type. Exceptions may be checked as well as unchecked. Errors happen at run-time. They will not be known to the compiler. Checked exceptions are known to compiler, whereas unchecked exceptions are not known to compiler because they occur at run-time. It is impossible to recover from errors. You can recover from exceptions by handling them through try-catch blocks. Errors are mostly caused by the environment in which the application is running. Exceptions are mainly caused by the application itself. Examples : java.lang.StackOverflowError, java.lang.OutOfMemoryError Examples : Checked Exceptions: SQLException, IOException Unchecked Exceptions: ArrayIndexOutOfBoundException, ClassCastException, NullPointerException

try-catch Statement

When you write a Java program, you come across exceptions. In order to catch these exceptions, you have a try-catch statement.

File Handling

The StandardOpenOptions Enums

WRITE – Opens the file for write access A

PPEND – Appends the new data to the end of the file

TRUNCATE\_EXISTING – Truncates the file to zero bytes

CREATE\_NEW – Creates a new file and throws an exception if the file already exists

CREATE – Opens the file if it exists or creates a new file if it does not

DELETE\_ON\_CLOSE – Deletes the file when the stream is closed

SPARSE – Hints that a newly created file will be sparse

SYNC – Keeps the file synchronized with the underlying storage device

Java Files

The File class provides many useful methods for creating and getting information about files.

Method Type Function

canRead() Boolean Tests the file if it is readable or not

canWrite() Boolean Tests the file if it is writable or not

createNewFile() Boolean Creates an empty file

delete() Boolean Deletes a file

exists() Boolean Tests the file if it exists

getName() String Returns the name of the file

getAbsolutePath() String Returns the absolute pathname

length() Long Returns the file size in bytes

list() String Returns an array of the files in the directory

mkdir() Boolean Creates a directory

Classes and Objects

A class is a building block and a template that describes the data and behavior associated with instances of a class. An entity that has a state and behavior is known as an object.

Encapsulation

Encapsulation is one the fundamental OOP concepts. It is a process of wrapping code and data together into a single entity.

Encapsulation can be achieved by: • Declaring the variables of a class as private. • Providing public setter and getter methods to modify and view the variable values.

Advantages of Encapsulation: • It improves maintainability, flexibility, and reusability. • The fields can be made read-only or write-only.

Abstraction

Abstraction is a process of hiding the implementation details from users. Only the functionality is provided to the user.

Abstract class is a class that is declared with abstract keyword. • Any class that contains more than one abstract method must be declared with abstract keyword. • Abstract classes can’t be directly instantiated with the new operator. • An abstract class can have parameterized constructors. The default constructor is present in an abstract class.

Polymorphism

Polymorphism is the ability of an object to perform a single action in different ways. The common use of polymorphism in OOP occurs when a parent class is used to refer to a child class object.

Runtime Polymorphism

It is achieved by virtual functions and pointers. • It is more flexible as all programs execute at run time.

Compile-time Polymorphism

It is achieved by function overloading and operator overloading. • Less flexible as all the programs execute at compile time.

Inheritance

Inheritance is a mechanism in which one class acquires all the properties and behaviors of the parent class. It can be used for method overriding and code reusability.

Association, Aggregation, and Composition

Association is the relationship between multiple objects.

Dependency: Aggregation implies a relationship where the child can exist independently of the parent. Composition implies a relation where the child cannot exist independently of the parent. • Type of Relationship: Aggregation is a “has-a” relation, whereas composition is a “part-of” relation. • Type of Association: Composition is a strong association where as aggregation is a weak association.

Data Structures

Data structures are specialized ways to store and retrieve the data. General data structure types are arrays, stacks, queues, and trees. Data structures are used to gather information from the user to process, maintain, and retrieve the information. Appropriate data structures help the code run smoothly.

Linked List

The linked list is a collection of nodes that are not stored at contiguous memory locations.

• The last node of the linked list contains the pointer to NULL. • The empty nodes cannot be present in the linked list. • The memory size of the linked list is limited and does not need to be declared in advance.

Singly Linked List

Every node contains two fields: data field and pointer to the next node.

It can move only in one direction.

It is the most commonly used linked list.

It is the collection of an ordered set of elements.

Circular Linked List

In this type of a linked list, the last node of the list has the pointer to the first node.

• It does not have a beginning or an end.

• Circular links go around the nodes until the program reaches the same node from where it started.

Doubly Linked List

• In a doubly linked list, each node contains the pointer to the previous as well as to the next node in the sequence. • In this type of a linked list, the pointer can move in both the directions.

Stack

Stack is a linear data structure in which insertion and deletion of elements can be done only at one end called the top.

It has a pointer to its top element

It works in a First In Last Out (FILO) manner.

Example: Coins stacked over one another

Top Value Stack Status

-1 The stack is empty.

0 There is only one element in the stack.

N-1 The stack is full.

N The stack gets overflowed.

Queue

A queue is a linear structure which enables the user to insert the elements from the REAR end and delete from the FRONT end.

It works in a First In First Out (FIFO) manner.

The difference between the stack and the queue is the way elements are removed.

Example: People waiting in a queue to vote.

Applications of the Queue To handle interrupts in operating systems To add or remove the songs in a playlist To arrange the data in a sequence in an MP3 or a CD player To assign waiting lists for single-shred resources like a printer or a disk To achieve synchronization in CPUs

Linear Search

Linear search is a simple algorithm. It can be used to search an element or a value in a given array by going through it from the beginning till the element is found.

Approach to linear searching: • Start from the leftmost element of an array. • Compare the element, say X, with each element of the array. • If the X matches with an element, return the index. • If the X does not match with any of the elements, return -1.

Binary Search

Binary search is also referred to as the logarithmic search. It finds the position of a particular value within a sorted array.

Approach to binary searching: • Compare an element, say X, with the middle element. • If X matches the middle element, return the mid index. • If X is greater than the mid element, it can lie only in the right half of the subarray after the mid element. • Otherwise, X can recur for the left half of the array, assuming the value is smaller than the mid element.

Exponential Search

Exponential search is also called the doubling search. It helps in searching through a sorted and unbounded array list for a specified input value. ● The first step determines a range in which the search key will reside if it is on the list. The second step performs the binary search on the array.

Selection Sort

Selection sort algorithm sorts an array by repeatedly finding the smallest number from the unsorted part and appending it at the beginning.

Bubble Sort

Bubble sort or sinking sort, is the simplest sorting algorithm. It works by repeatedly swapping the adjacent elements if they are in an incorrect order.

Insertion Sort

Insertion sort builds the final sorted array by moving the elements to a sorted sublist. It is less efficient for a large set of values.

Quick Sort

Quick sort is a divide and conquer algorithm. It chooses an element as a pivot and divides the given array around the chosen pivot.