raisoni EDUCATION

EXPERIMENT NO. 1.

Aim:-

Setting up Java Development Environment.

- Install JDK and setup the development environment
- Write & execute a simple Java program to print a string
- Explore basic structure of a Java program, including data types, variables & operators.

Theory:-

- Install JDK & setup development environment.

To begin developing Java applications, it's essential to set up Java Development Environment

1) Java Development Kit JDK:-

The JDK is a software development kit used to develop Java applications. It includes foll. components:-

- Java Runtime Environment (JRKs).
- Java Compiler (Javac)
- Java Virtual Machine (JVM).

raisoni EDUCATION

2. Steps to Install:-
   1. Download JDK.
   2. Install JDK.

3. Setting Up Environment Variables:-
   Go to "system properties" → "Advanced system" → "Environment variables". Under 'system variable", click 'New' & add variable
   - variable name : Java_home.

4. Verifying installation:-
- Open command Prompt or Terminal
- Type java -version to check installed version
- Type javac --version to check version of compiler
- If both commands return installed version, setup is complete.

5. Setting up a Development Environment:-
   Popular Java IDEs:-
   - Eclipse
   - IntelliJ IDEA
   - NetBeans.

Conclusion:-
Thus, we have understood Java Development Environment & implemented a program of it.

Seen

Experiment NO. 2

Aim:-

Control flow & Arrays.

- Implement programs using control flow statements (if-else, switch, loops).
- Create & manipulate array, including multidimensional arrays.
- Write a program to sort an array using different sorting techniques.

Theory:-

Control flow statement using flow statements such as if-else, switch & loops.

1. Control flow statements:

- If-else statement:

The if-else allows a program to certain code based on a condition

Syntax:-

```
If (condition) {
    // code to execute if condition is true
} else {
    // code to execute if condition is false
}
```

Switch statement:-
It tests value of a variable & execute different blocks of code depending on value.
Syntax:-

```
switch (expression) {
    case value1:
        // code
    break;
    case value2:
        // code
        break;
    default:
        // code to execute if no case matches.
    break;
}
```

Loops:- 1) for loops:
It is used when number of iterations is known
Syntax:-

```
for (initialization; condition; update) {
    // code

}
```

2) while loop:-
Used when number of iterations is not known beforehand

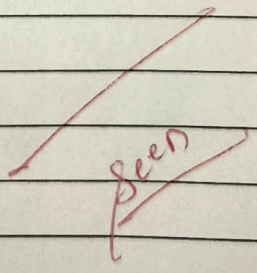
```
while (condition) {
        //code
}
```

3) Do-while:-
~~Arrays~~ It is a similar to while loop, but it executes code at least once before checking condition

```
do {
    //code
} while (condition);
```

Conclusion:-

Thus, we have successfully implemented program.



raisoni EDUCATION

Experiment No. 3.

Aim :-

Object-Oriented programming concepts.
- Implement create classes & objects in Java
- Implement constructors, methods, and method overloading.
- Write a program to ~~sort~~ demonstrate inheritance & method overriding

Theory :-

In OOPs, a class is a blueprint for creating objects.
A class defines properties & behaviours that object created ~~fo~~ from it will have.

- Class :-
A class contains fields (variable) and methods (functions) that define behaviour of objects.

- Object :-
An object is an instance of class. When class is defined, no memory is allocated until an object is created.

raisoni EDUCATION

Syntax for class:-

```
class className {
    // Fields (attributes)
    int attribute1;
    String attribute2;

    // Methods
    void method1() {
        // code
    }
}
```

Syntax for Object:-

```
className obj = new className();
obj.method1();
```

Conclusion:-

Thus, we have implemented program successfully.

Seen

raisoni EDUCATION

Experiment No. 4

Aim:-

Abstract classes & Interfaces.

- Implement a program using abstract classes & interfaces.
- Demonstrate polymorphism using dynamic method dispatch.
- Create packages & import them into main program

Theory:-

Abstract Classes.

An abstract class is a class that cannot be instantiated directly. It can contain abstract methods (methods without body), as well as concrete method (methods with body).

Abstract class:- A class that contains one or more abstract methods.

- Key features:
- can have both abstract methods & methods with implementation
- Abstract methods must be implemented by subclasses.
- cannot be instantiated directly.

2 Interfaces:-

It is a completely "abstract class" that is used to group related methods with empty bodies. It defines a contract that implementing classes must follow.

Key features

- All methods are abstract & public
- A class can implement multiple interfaces, overcoming single inheritance limitation.
- can contain default & static methods

- Polymorphism :- allows objects to be treated as instances of their parent class. Dynamic methods is a mechanism by which a call to an overriden method is resolved at runtime, rather than compile time.

- Packages are used to group related classes & interfaces together.

Conclusion:-

Thus we have successfully implemented the program.

Seen

raisoni EDUCATION

Experiment No. 5

Aim:-

Exception Handling.

Write a program to handle exceptions using try, catch, finally, throw, & throws. Implement custom exceptions. Demonstrate the use of multiple catch blocks & nested try statements.

Theory:-

It is a powerful mechanism that handles runtime errors, allowing program to continue execution rather than terminating unexpectedly.

Try, catch, finally blocks:-

Try block:- Code that might throw an expection is placed inside a try block

Catch block:- It catches exceptions thrown by the try block. You can have multiple catch blocks.

Finally block:- It will always execute after try & catch blocks, regardless of whether an exception was thrown.

raisoni EDUCATION

Syntax:-

```
try {
    // code that may throw exception
} catch (ExceptionType1 e1) { //code
//code (ExceptionType2 e2) {
    //code to handle exception1
} finally {
    //code will execute always
}
```

* Throwing Exceptions:-

- You can use ~~ne~~ throw keyword to explicitly throw an exception from your code.

Syntax:-

```
throw new ExceptionType ("Error message");
```

- Throws:- It indicates that method can throw one or more exceptions.

Syntax:-

```
public returnType methodName() throws ExceptionType1,
    ExceptionType2 {
// Method code
}
```

Conclusion:-

Thus, we successfully implemented Exception Handling in Java which is a crucial aspect of writing robust programs.

Seen

raisoni EDUCATION

Experiment No. 6

Aim:-

Multithreading :-

It is a powerful feature in Java that allows concu

- Create a method & execute threads by extending Thread class & implementing Runnable interface
- Implement synchronization & inter-thread communication.
- Write a program to demonstrate the producer-consumer problem

Theory :-

Multithreading a powerful feature in Java that allows concurrent execution of two or more threads.

Creating Threads by extending the Thread class, you need to :-

- Create a new class that extends the Thread class.
- Override the run() method to define the code that should be executed by thread.
- Create an instance of your thread class & call start() method.

Synchronization:-
It is a crucial in multithreading especially when multiple threads access shared resources.
It ensures that only thread can access a resource at a time, preventing data inconsistency.

Inter-thread communication:
It is essential for coordinating actions between threads, especially when one thread needs to wait for another to complete.

Producer:- This thread generates data & places it into a shared buffer (queue). If the buffer is full, the producer wait until space becomes available.

Consumer:- This thread takes data from shared buffer. If buffer is empty, consumer waits until data is available.

Conclusion:-
Thus we successfully implemented producer consumer problem & understood multi-threading.

Experiment NO.07

Aim:

Java collections Framework.
Use various collection classes (List, Set, Map) and their methods.
Write a program to perform operations on collection, such as adding, iterating, and sorting element.
Implement generic class & demonstrate it use.

Theory:-

The Java collections Framework (JCF) is a unified architecture for storing & manipulating groups of objects.
It includes various interfaces, classes and algorithms for handling collections. such as lists, sets & map.

Key components of Java collections framework.

1) List interface:-
- List is an ordered collection that allows duplicate elements.
- Commonly used classes that implement the List interface include ArrayList, LinkedList, & Vector.

2) Set Interface:-
- Set is a collection that does not allow duplicate elements & is unordered
- used in HashSet, LinkedHashSet, & TreeSet.
- common methods : add(), remove(), iterator().

3) Map Interface:-
- Map is a collection that maps keys to values.
- classes include HashMap, LinkedHashmap.

Generic class:-
Generic allow types to be parameters when defining classes, interfaces and methods. This provides a way to create classes that are flexible & can work with any type of data, ensuring type safety at compile time.

Conclusions:-
Thus, we have understood the Java collection frameworks & implemented code successfully.

Experiment No. 8

Aim:-

File Handling & Serialization.

- Perform file input & output operations.
- Write a program to read from & write to a text file.
- Implement serialization & deserialization of objects.

Theory:-

1) File Handling in Java.

→ can be done using classes like File, FileReader, FileWriter, BufferedReader, etc.

- File Handling allow Java programs to read & write data to & from files on local file system.

- File class:-
  This represents file paths, allowing program to interact with files & directories

- FileReader & FileWriter:-
  These are character-based classes for reading & writing text files.
  FileReader reads contents of file,
  filewriter writes data to a file.

Serialization & Deserialization:-

1] Serialization:-
This is the process of converting an object into a byte stream so that it can be saved to file or sent over a network.

2] Deserialization:-
This is reverse process of reading the byte stream and converting it back into an object.

Conclusion

Thus, we have successfully implemented program of file Handling.

9. Experiment NO.9

Aim:-
Creating Basic layouts with XML.

- Install Android studio & set up development environment.
- Create an Android project & design simple UI layouts using XML.
- Implement various views & view Groups.

Theory:-

Android studio allows developers to design User Interfaces (UIs) using XML. The layout of an Android application's UI is described in XML files, which makes it easier to design interfaces separately from application logic

1) View:-

- Text View : Displays text to user
- Button :- A clickable element, often used for user actions.
- Image View :- Displays an image.

2) View Groups.

- Linear Layout:-
  Arranges its children in a single direction - either vertically or horizontally.

- Relative Layout:-
  Positions its children relative to each other or relative to parent container.

The layout is defined in an XML file (activity_main.xml), while interaction (e.g. Button clicks) is handled in Java code (MainActivity.java).

Conclusion:-
Thus, we have understood about XML Basic UI layouts & implemented it.

Seen

Experiment No. 10

Aim:-

Advanced Layout Design

- Design responsive layouts using XML
- Use styles & themes to customize UI elements.
- Implement drawable resources & animations.

Theory:-

Responsive Design ensures that your app adapts to different screen sizes and orientations.
In Android, this is achieved using flexible layouts like ConstraintLayout, LinearLayout & RelativeLayout, along with qualifiers for different layouts.

Key concepts:-

1) Constraint layout:-

Allows flexible positioning of UI elements relative to each other & parent.

Size Qualifiers:-
Create different layouts for different screen sizes (res/layout-large, etc).

3) Drawable Resources:-

It defines shapes, gradients, or images that can be applied to UI elements like buttons or backgrounds

Android provides ShapeDrawable and BitmapDrawable to design vector & image-based graphics

4) Animations:-
It makes your app feel dynamic. Android provides Animator & Animation classes for creating property animations & view animations

Conclusion:-

Thus we have successfully implemented program.

Seen