draw.io

https://plantumlonlineeditor.com/

1. Program using class, object, package, overloaded methods, command line arguments

Folder structure:

```
src/
└── com/
    └── scsvmv/
        └── javalab/
            ├── Hello.java
            ├── TestHello.java
            └── hello/
                └── Hello.java
```

Files:

- com/scsvmv/javalab/Hello.java
  → Simple Hello class (always prints *Hello World*)

- com/scsvmv/javalab/TestHello.java
  → Main class with command-line arguments

- com/scsvmv/javalab/hello/Hello.java
  → Overloaded wish() methods and Date usage

Steps to Compile the Program

1. Open Command Prompt / Terminal
2. Navigate to the src folder using cd command
   Note: Always compile from the **src folder**, not inside package folders.
3. Compile all Java files

…..src>javac com/scsvmv/javalab/hello/Hello.java
…..src>javac com/scsvmv/javalab/Hello.java
…..src>javac com/scsvmv/javalab/TestHello.java

✔ .class files will be generated in the same package folders.

Steps to Run the Program
Run with command-line argument:
java com.scsvmv.javalab.TestHello Seetharaman

If no argument is passed:
java com.scsvmv.javalab.TestHello

Sample Output (with argument)
Hello World
Hello Seetharaman
Hello Seetharaman | Date: Thu Feb 05 10:30:15 IST 2026

Output (without argument)

Hello World
Usage:
java com.scsvmv.javalab.TestHello <name>
Example:
java com.scsvmv.javalab.TestHello Seetharaman

TestHello.java

```
package com.scsvmv.javalab;

import com.scsvmv.javalab.hello.Hello;

/*
 * TestHello class
 * Demonstrates passing command-line arguments
 */
public class TestHello {

   public static void main(String[] args) {

      // Using Hello class from parent package
      Hello simpleHello = new Hello();
      simpleHello.wish();

      // Using Hello class from child package
      com.scsvmv.javalab.hello.Hello h =
            new com.scsvmv.javalab.hello.Hello();

      // Check for command-line argument
      if (args.length > 0) {
         h.wish(args[0]);          // pass CLI argument
         h.wishWithDate(args[0]);
      } else {
         System.out.println("Usage:");
         System.out.println("java com.scsvmv.javalab.TestHello <name>");
         System.out.println("Example:");
         System.out.println("java com.scsvmv.javalab.TestHello Student");
      }
   }
}
```

Hello.java (com.scsvmv.javalab**)**

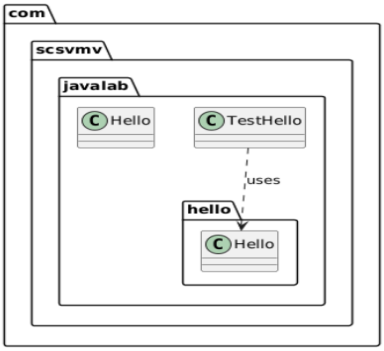
```
package com.scsvmv.javalab;

/*
 * Simple Hello class
 * Always wishes Hello World
 */
public class Hello {

   public void wish() {
```

```
        System.out.println("Hello World");
    }
}
```

Hello.java (com.scsvmv.javalab.hello)

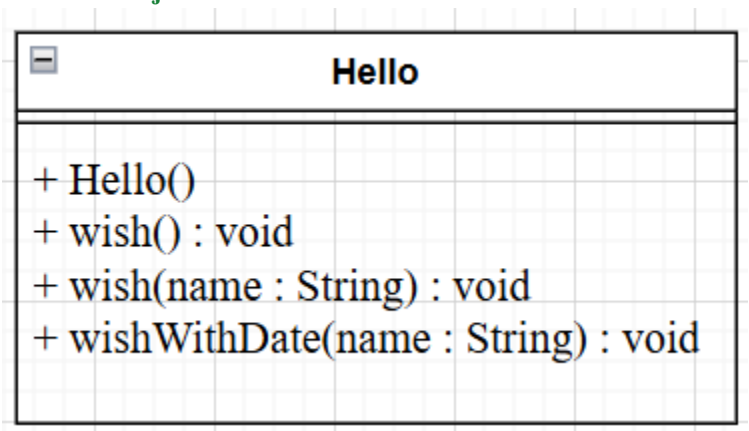
```
package com.scsvmv.javalab.hello;

import java.util.Date;

/*
 * Hello class
 * Demonstrates method overloading and command-line arguments
 */
public class Hello {

    // Default constructor
    public Hello() {
    }

    // Generic wish
    public void wish() {
        System.out.println("Hello World");
    }

    // Overloaded wish method with name parameter
    public void wish(String name) {
        System.out.println("Hello " + name);
    }

    // Wish with name and today's date
    public void wishWithDate(String name) {
        Date today = new Date();
        System.out.println("Hello " + name + " | Date: " + today);
    }
}
```

Package Diagram

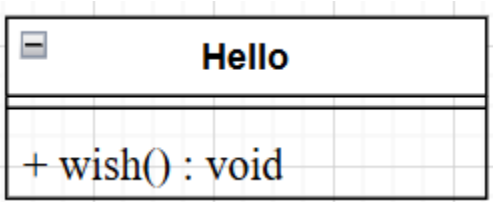


Class Diagrams

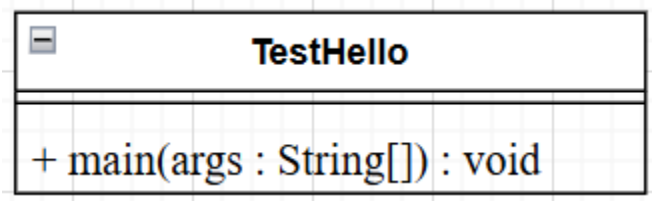
com.scsvmv.javalab.hello.Hello



com.scsvmv.javalab.Hello



com.scsvmv.javalab.TestHello



Viva Questions: Basic Level

1. What is a package in Java? Why do we use packages in your program?
2. What is the role of the Hello class in com.scsvmv.javalab and in other package ?
3. How does the TestHello class instantiate objects of Hello?
4. What are overloaded methods? Which methods are overloaded in your program?
5. How do you pass command-line arguments to the wish method?
6. What is the difference between Hello in the parent package and Hello in the child package?

Viva Questions: Intermediate Level

1. How does the wishWithDate() method work? Which Java class does it use?
2. What happens if no command-line argument is provided in TestHello?
3. How do you compile and run your program considering the package structure?
4. Can com.scsvmv.javalab.hello.Hello access private members of com.scsvmv.javalab.Hello? Why or why not?

Viva Questions: Advanced Level

1. How would you modify the program to print the current date and time in a specific format (like dd-MM-yyyy HH:mm:ss)?
2. If multiple developers are working on this program, how does package separation help in project organization?

3. Explain the difference between overloaded constructor vs overloaded method in terms of design and usability in your program.
4. How would you convert this program into a jar and execute it from the command line?
5. Discuss how UML diagrams can help in understanding and maintaining this program.

Extra Practical / Trick Questions

1. What happens if you write Hello h = new Hello(); in TestHello without importing the child package class?
2. Can you have two classes with the same name in the same package? Why or why not?
3. How do you handle multiple overloaded methods with the same parameters but different return types?
4. Why is it important to pass command-line arguments to wish() instead of hardcoding the name?

Prog2: Program to explore java.lang.Math class methods and command line arguments, custom package

```
project-folder/
│
├── Enter2Float.java
└── operations/
    └── BasicOperations.java
```

operations/BasicOperations.java

```
package operations;

public class BasicOperations {

   private float num1;
   private float num2;

   // Constructor for float values
   public BasicOperations(float num1, float num2) {
      this.num1 = num1;
      this.num2 = num2;
   }

   // Overloaded constructor for int values
   public BasicOperations(int num1, int num2) {
      // Convert ints to float
      this.num1 = (float) num1;
      this.num2 = (float) num2;
   }

   public void performOperations() {

      System.out.println("--- Basic Arithmetic ---");
      System.out.println("Addition: " + (num1 + num2));
      System.out.println("Subtraction: " + (num1 - num2));
      System.out.println("Multiplication: " + (num1 * num2));

      if (num2 != 0) {
```

```
            System.out.println("Division: " + (num1 / num2));
            System.out.println("Modulus: " + (num1 % num2));
        } else {
            System.out.println("Division and Modulus: Cannot divide by zero");
        }

        System.out.println("\n--- Math Class Methods ---");
        System.out.println("Absolute value of num1: " + Math.abs(num1));
        System.out.println("Maximum: " + Math.max(num1, num2));
        System.out.println("Minimum: " + Math.min(num1, num2));
        System.out.println("Power (num1^num2): " + Math.pow(num1, num2));
        System.out.println("Square root of num1: " + Math.sqrt(num1));
        System.out.println("Rounded num1: " + Math.round(num1));
        System.out.println("Floor of num1: " + Math.floor(num1));
        System.out.println("Ceil of num1: " + Math.ceil(num1));
        System.out.println("Random number (0-1): " + Math.random());
    }
}
```

Enter2Float.java

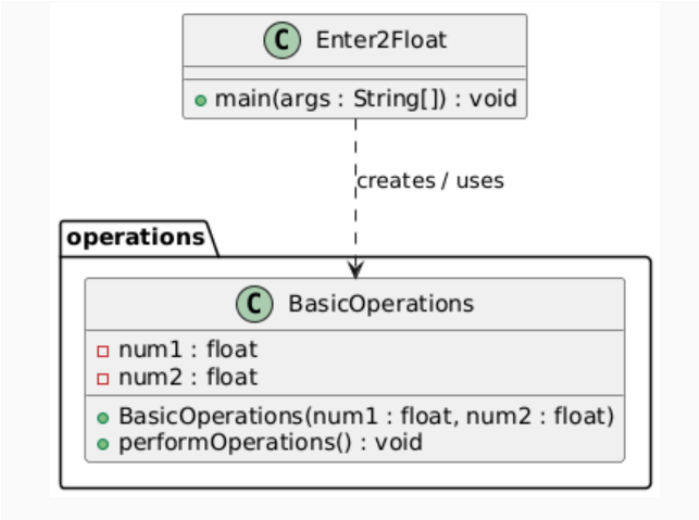
```
import operations.BasicOperations;

public class Enter2Float {

    public static void main(String[] args) {

        if (args.length != 2) {
            System.out.println("Usage: java Enter2Float <num1> <num2>");
            return;
        }

        try {
            float num1 = Float.parseFloat(args[0]);
            float num2 = Float.parseFloat(args[1]);

// Using float constructor
BasicOperations obj1 = new BasicOperations(num1, num2);
obj1.performOperations();

// Using int constructor
BasicOperations obj2 = new BasicOperations(10, 3);
obj2.performOperations();

        } catch (NumberFormatException e) {
            System.out.println("Please enter valid float numbers.");
        }
    }
}
```

Compile & Run
Open CMD inside project-folder and run:

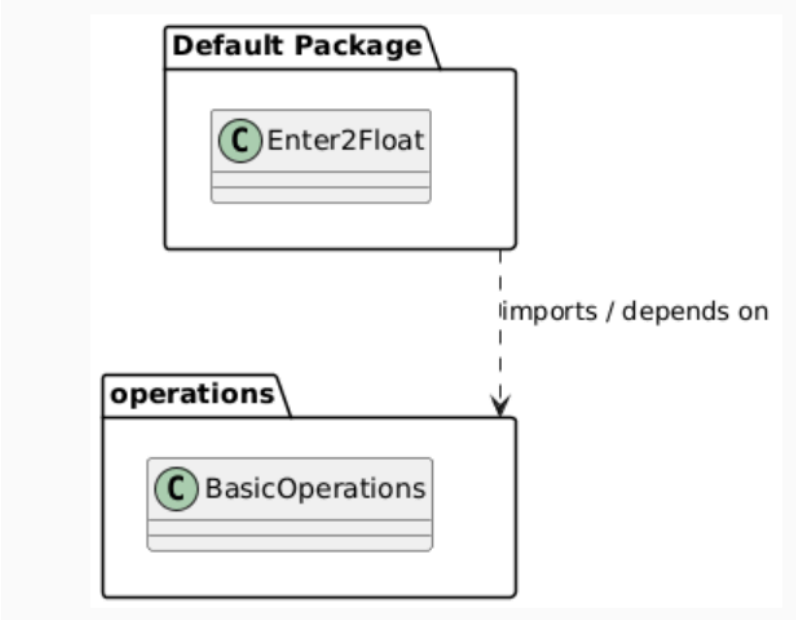
javac operations/BasicOperations.java Enter2Float.java

java Enter2Float 10.5 3.2

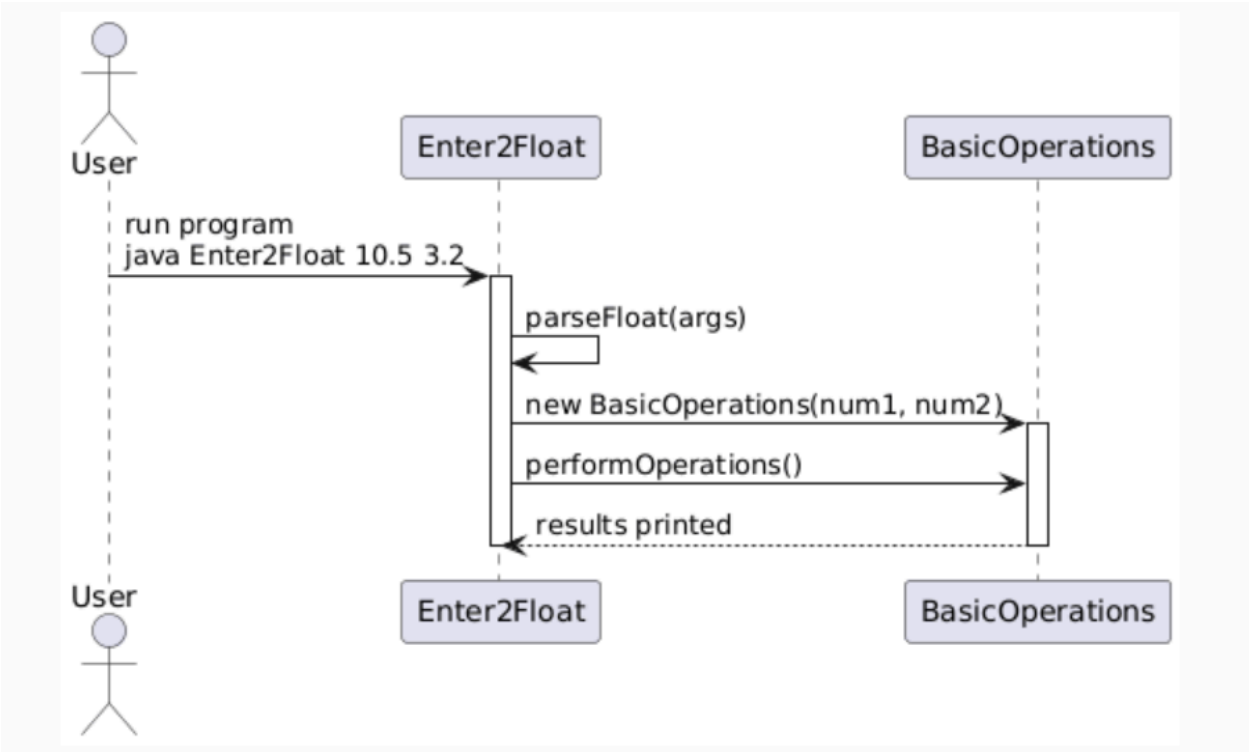
UML Class Diagram



Package Diagram



Sequence Diagram



Pgm3: To explore String and StringBuffer methods

```
project-folder/
├── Enter2Strings.java
└── textops/
    ├── StringOperations.java
    └── StringBufferOperations.java
```

textops.StringOperations.java

```
package textops;

public class StringOperations {

    private String str1;
    private String str2;

    public StringOperations(String str1, String str2) {
        this.str1 = str1;
        this.str2 = str2;
    }

    public void exploreStringMethods() {
        System.out.println("----- String Methods -----");

        System.out.println("Length of str1: " + str1.length());
```

```
        System.out.println("Uppercase str1: " + str1.toUpperCase());
        System.out.println("Lowercase str1: " + str1.toLowerCase());
        System.out.println("Concatenation: " + str1.concat(str2));
        System.out.println("Equals: " + str1.equals(str2));
        System.out.println("Equals Ignore Case: " + str1.equalsIgnoreCase(str2));
        System.out.println("Contains str2 in str1: " + str1.contains(str2));
        System.out.println("Substring of str1 (0,3): " +
                (str1.length() >= 3 ? str1.substring(0, 3) : "Too short"));
        System.out.println("Replace 'a' with 'x' in str1: " + str1.replace('a', 'x'));
        System.out.println("Index of str2 in str1: " + str1.indexOf(str2));
        System.out.println("Trim str1: '" + str1.trim() + "'");
    }
}
```

textops.StringBufferOperations.java

```
package textops;

public class StringBufferOperations {

    private String str;

    public StringBufferOperations(String str) {
        this.str = str;
    }

    public void exploreStringBufferMethods(String appendStr) {
        System.out.println("\n----- StringBuffer Methods -----");

        // Create a StringBuffer object from str
        StringBuffer sb = new StringBuffer(str);

        System.out.println("Original StringBuffer: " + sb);

        // append() adds text to the end of the buffer (mutable)
        sb.append(appendStr);
        System.out.println("After append: " + sb);

        // insert() inserts text at a specified index (mutable)
        sb.insert(0, "Start-");
        System.out.println("After insert: " + sb);

        // replace() replaces characters between start and end index (mutable)
        sb.replace(0, 5, "Begin");
        System.out.println("After replace: " + sb);

        // delete() removes characters between start and end index (mutable)
        sb.delete(0, 6);
        System.out.println("After delete: " + sb);

        // reverse() reverses the buffer content (mutable)
        sb.reverse();
```

```
        System.out.println("After reverse: " + sb);

        // capacity() shows current buffer size; it automatically increases as needed
        System.out.println("Capacity: " + sb.capacity());

        // length() shows the number of characters currently in the buffer
        System.out.println("Length: " + sb.length());

        // NOTE: Unlike String, these operations modify the same object
        // String operations (concat, replace, etc.) return new String objects
    }
}
```

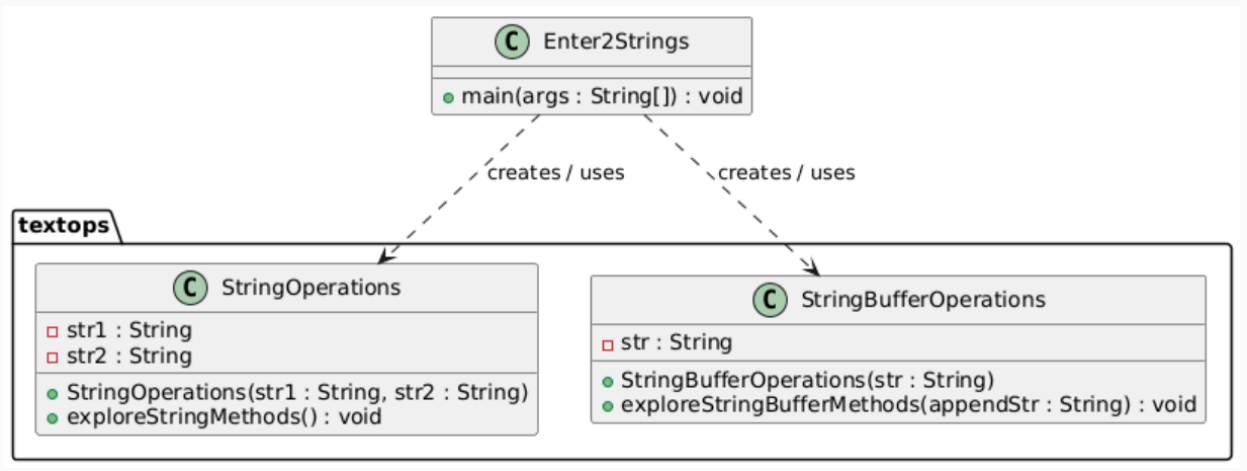
Enter2Strings.java

```
import textops.StringOperations;
import textops.StringBufferOperations;

public class Enter2Strings {

    public static void main(String[] args) {

        if (args.length != 2) {
            System.out.println("Usage: java Enter2Strings <str1> <str2>");
            return;
        }

        String str1 = args[0];
        String str2 = args[1];

        // Instantiate and call StringOperations
        StringOperations stringOps = new StringOperations(str1, str2);
        stringOps.exploreStringMethods();

        // Instantiate and call StringBufferOperations
        StringBufferOperations sbOps = new StringBufferOperations(str1);
        sbOps.exploreStringBufferMethods(str2);
    }
}
```

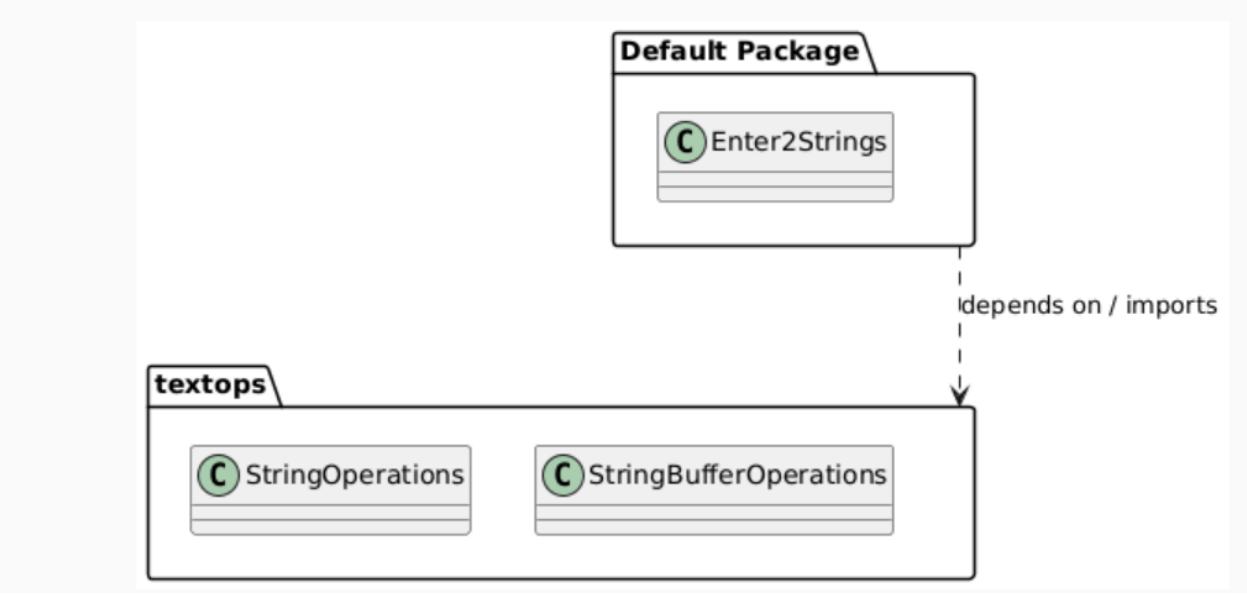
Compile & Run

```
proj-folder>javac textops/StringOperations.java textops/StringBufferOperations.java Enter2Strings.java

proj-folder>java Enter2Strings Hello World
```

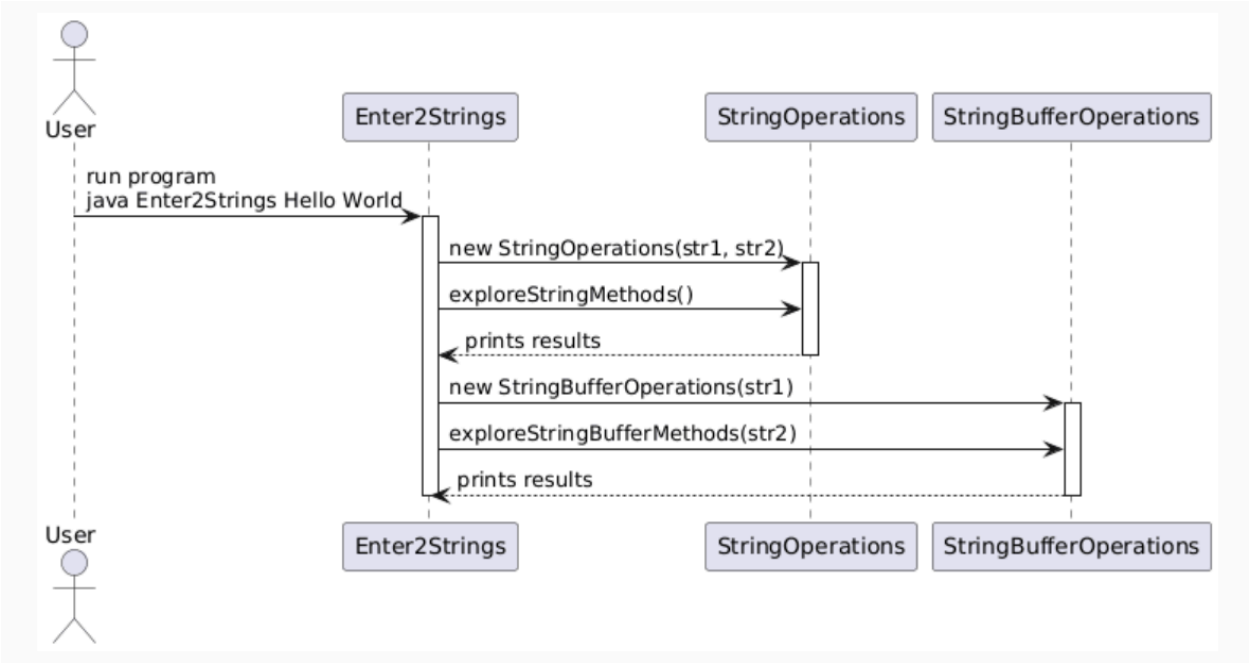
UML Diagrams:
Class Diagram



Package Diagram



Sequence Diagram



Questions

1 Basic Java / Syntax

1. What is the difference between float and double in Java?
2. Why did we use Float.parseFloat(args[0]) instead of directly assigning args[0]?
3. Can Math class methods be called on objects? Why or why not?
4. What happens if args.length != 2? How does the program handle it?

---

2 Math Class

5. Name 5 commonly used methods of the Math class and their purpose.
6. What is the difference between Math.round(), Math.ceil(), and Math.floor()?
7. How is Math.random() generated, and what is its range?
8. What does Math.pow(a, b) do if b is negative?
9. Can Math.sqrt(-1) be called? What happens?

---

3 OOP Concepts

10. Why did we move operations to a separate class (BasicOperations)?
11. What is the relationship between Enter2Float and BasicOperations in UML?
12. Why do we use a constructor in BasicOperations?
13. What is dependency, and where is it used in this program?

---

4 Command-Line Arguments

14. How do you pass command-line arguments when running a Java program?
15. What type of exception can occur when parsing a non-numeric string to float?
16. Can you run the program without arguments? What will happen?

Enter2Strings (String/StringBuffer Operations)

1 Basic Java / String

1. What is the difference between String and StringBuffer?
2. Why is String immutable?
3. Which methods of String create a new object and which modify the existing one?

4. What happens if you call substring() with an invalid range?

---

2 String Methods

5. What is the difference between equals() and == for strings?
6. Explain equalsIgnoreCase() with an example.
7. What is the difference between trim() and strip()?
8. What does indexOf() return if the substring is not found?
9. How does replace() differ from replaceAll()?

---

3 StringBuffer Methods

10. What makes StringBuffer faster for multiple string modifications compared to String?
11. Explain what append(), insert(), replace(), delete(), and reverse() do.
12. What is the difference between StringBuffer and StringBuilder?
13. What does capacity() mean in StringBuffer? How is it different from length()?

---

4 OOP / Package

15. Why did we create separate classes for String and StringBuffer operations?
16. Explain the UML dependency relationship between Enter2Strings and the operation classes.

---

5 Command-Line Arguments & Runtime

19. What happens if you pass fewer or more than 2 strings?
20. Could this program handle empty strings? What would happen?
