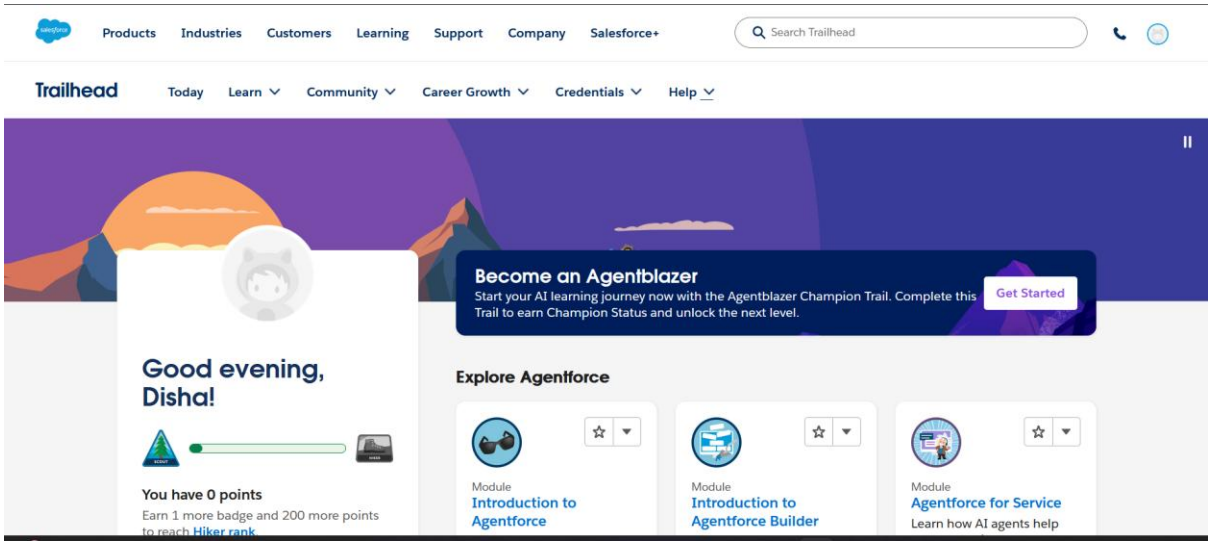
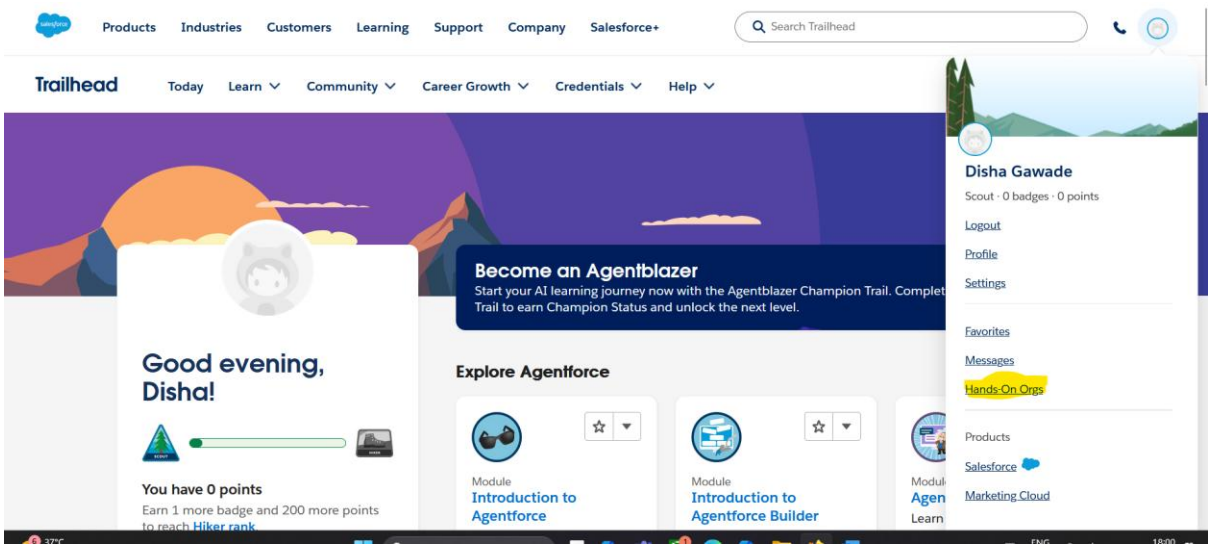
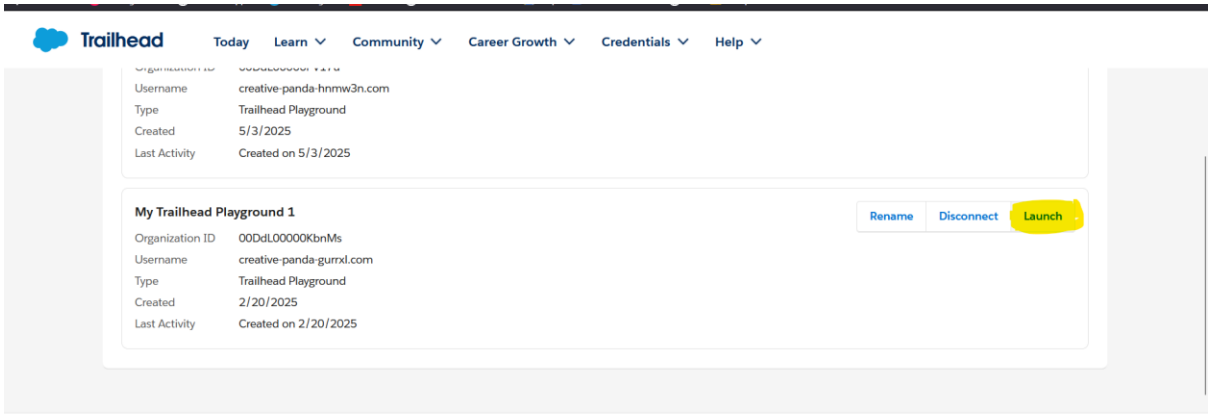
14.Creating an application in salesForce.com using Apex Programming Language .Perform Multiplication of Two numbers

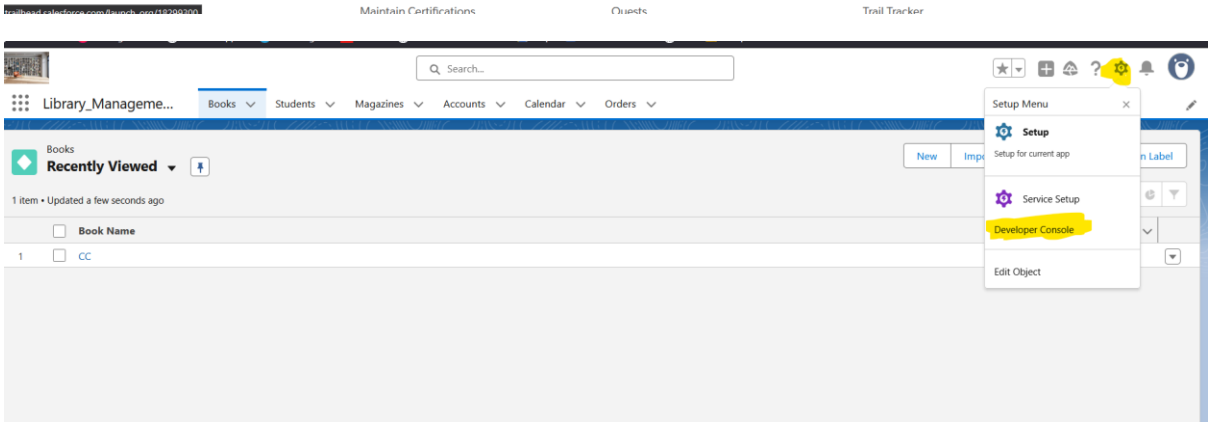
Step1 :- Create account and login

https://trailhead.salesforce.com/today

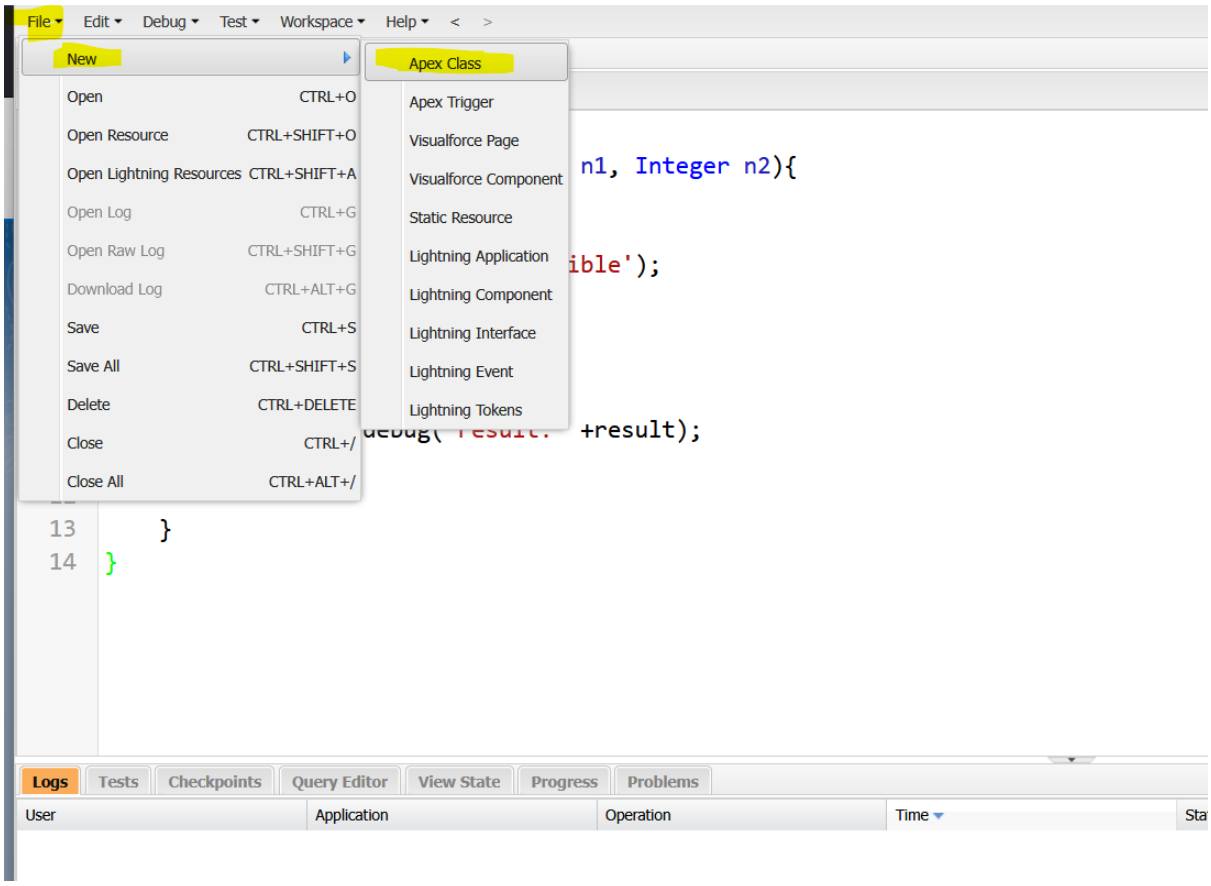





Trailhead
Today
Learn
Community
Career Growth
Credentials
Help
Username
creative-panda-hnmw3n.com
Type
Trailhead Playground
Created
5/3/2025
Last Activity
Created on 5/3/2025
My Trailhead Playground 1
Rename
Disconnect
Launch
Organization ID
00DdL00000KbnMs
Username
creative-panda-gurrxl.com
Type
Trailhead Playground
Created
2/20/2025
Last Activity
Created on 2/20/2025


Learn
Trails
Trailmixes
Credentials
Superbadges
Certifications
Community
Trailblazer Community
Events
Extras
Trailhead Login
Sales Enablement

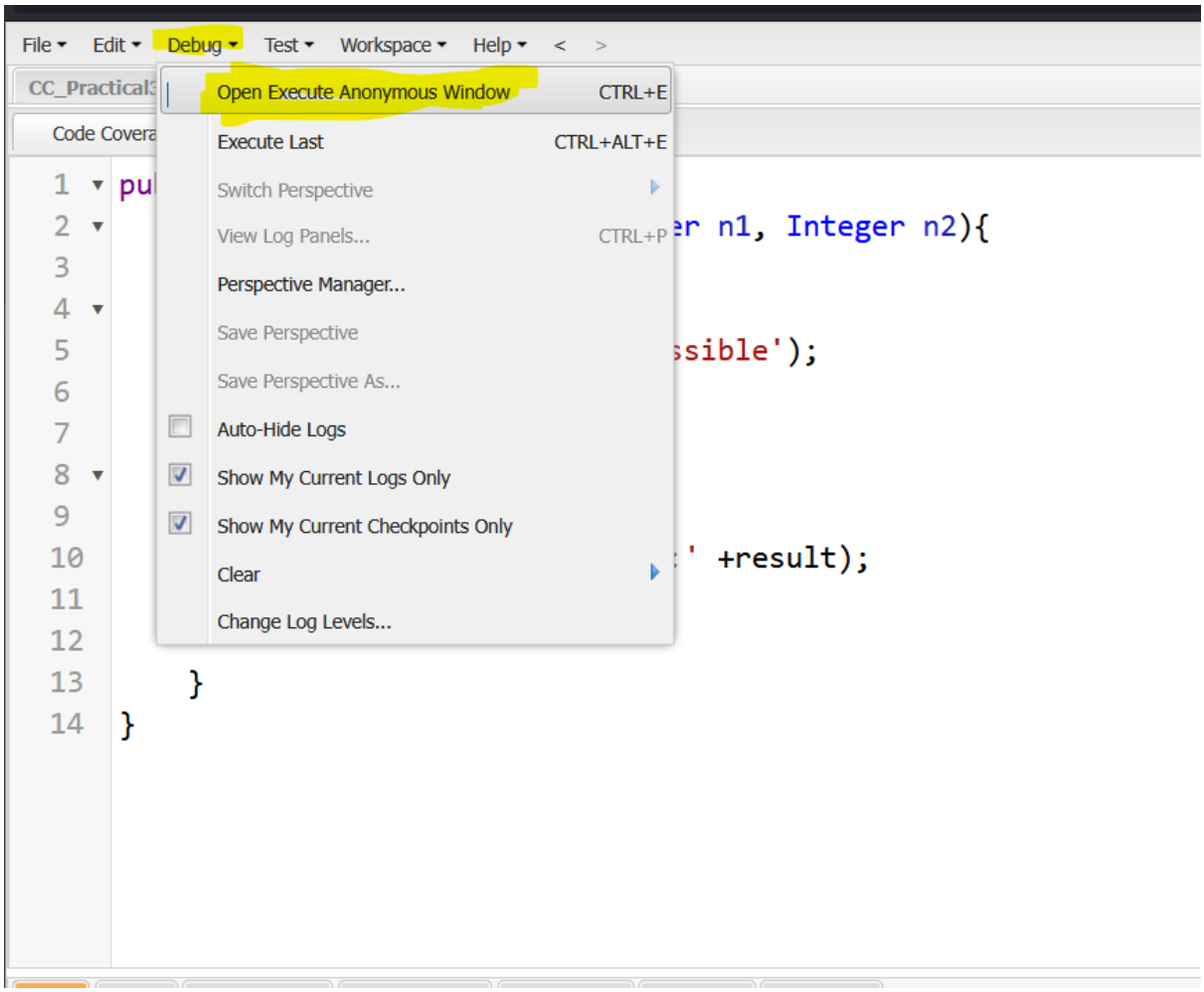
Library_Manageme...
Books
Students
Magazines
Accounts
Calendar
Orders
Recently Viewed
1 item • Updated a few seconds ago
Book Name
cc
Setup Menu
Setup
Setup for current app
Service Setup
Developer Console
Edit Object

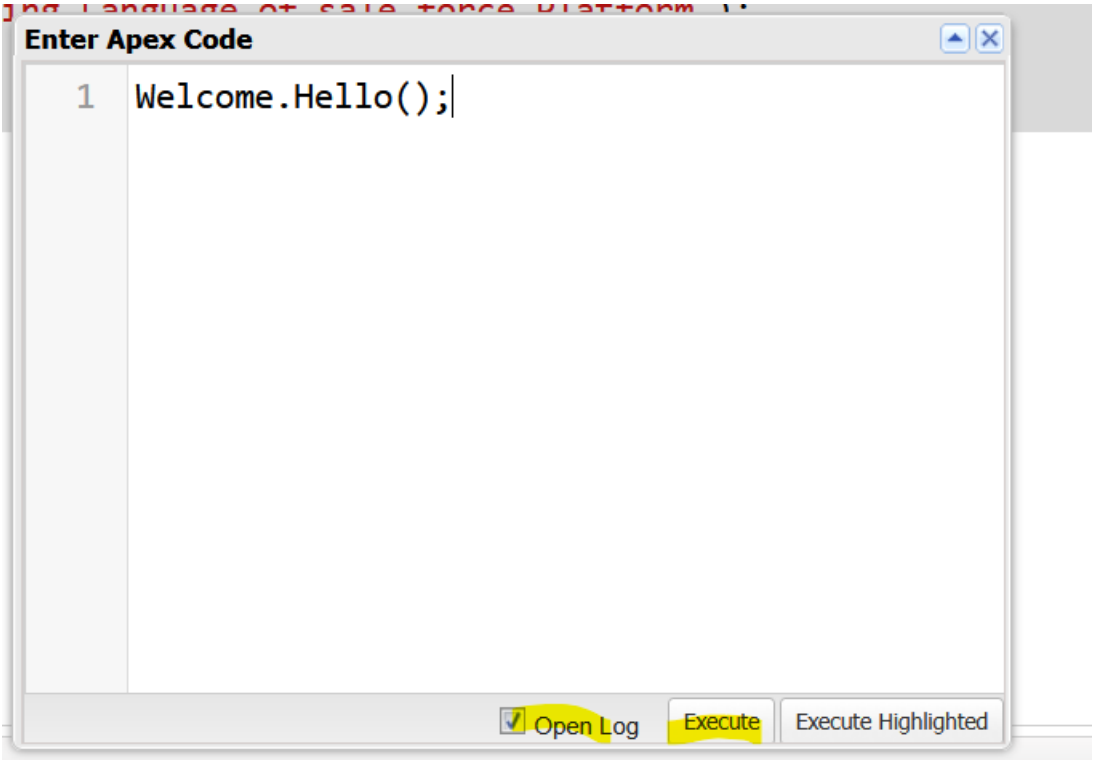

File
Edit
Debug
Test
Workspace
Help
New
Open CTRL+O
Open Resource CTRL+SHIFT+O
Open Lightning Resources CTRL+SHIFT+A
Open Log CTRL+G
Open Raw Log CTRL+SHIFT+G
Download Log CTRL+ALT+G
Save CTRL+S
Save All CTRL+SHIFT+S
Delete CTRL+DELETE
Close CTRL+/
Close All CTRL+ALT+/
Apex Class
Apex Trigger
Visualforce Page
Visualforce Component
Static Resource
Lightning Application
Lightning Component
Lightning Interface
Lightning Event
Lightning Tokens
n1, Integer n2){
ible');
+result);
13 }
14 }
Logs
Tests
Checkpoints
Query Editor
View State
Progress
Problems
User
Application
Operation
Time

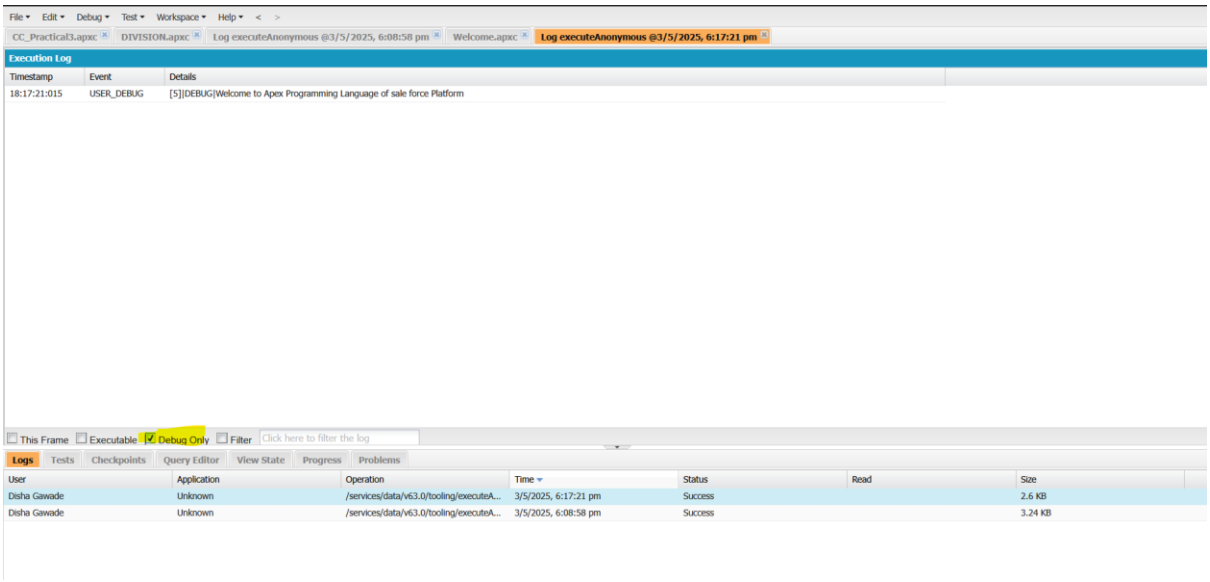
Create apex class and write this code

```
public class firstClass1 {


    public static void Multi()
    {
        Integer a = 4;
        Integer b = 5;
        Integer c = a * b;
        System.debug(c);
    }
}
```







Explanation of Concepts in the Code

1. Apex Class

- Concept : Apex classes are blueprints for objects and encapsulate data and behavior. They are fundamental in Salesforce development.

- Code : `public class firstClass1 { ... }`

- Details : This defines a class named `firstClass1` with `public` access (visible across namespaces).

2. Static Method

- Concept : A static method belongs to the class itself, not to instances of the class. It can be called without creating an object.
- Code : `public static void Multi() { ... }`
- Details : The `Multi` method is static, meaning it can be invoked directly via `firstClass1.Multi()`.

3. Variables and Data Types

- Concept : Apex uses strongly-typed variables. Here, `Integer` is used to store whole numbers.
- Code : `Integer a = 4; Integer b = 5;`
- Details : `a` and `b` are hardcoded integers. `c` stores the result of `a * b`.

4. Arithmetic Operations

- Concept : Apex supports arithmetic operators like `*` for multiplication.
- Code : `Integer c = a * b;`
- Details : Multiplies `a` and `b` and assigns the result to `c`.

5. Debugging in Apex

- Concept : `System.debug()` logs messages for debugging, visible in the Salesforce Developer Console.
- Code : `System.debug(c);`
- Details : Outputs the value of `c` (20) to debug logs.

Line-by-Line Explanation

```apex
public class firstClass1 { // Defines a public Apex class named firstClass1.
    public static void Multi() { // Declares a static method Multi() that returns nothing (void).
        Integer a = 4; // Initializes an Integer variable a with value 4.
        Integer b = 5; // Initializes an Integer variable b with value 5.
        Integer c = a * b; // Multiplies a and b, storing the result in c (4 * 5 = 20).
        System.debug(c); // Logs the value of c (20) to debug output.
    }
}
```

Potential Questions & Answers

1. Why is the method `Multi()` declared as `static`?

- Answer : Static methods can be called without instantiating the class. This is useful for utility methods like this one.

2. How would you make this code dynamic (e.g., accept user input)?

- Answer : Modify the method to accept parameters:

```apex
public static void Multi(Integer a, Integer b) {
    Integer c = a * b;
    System.debug(c);
}
```

3. Where can you see the output of `System.debug(c)`?

- Answer : In the Debug Logs of the Salesforce Developer Console.

4. What is the scope of variables `a`, `b`, and `c`?

- Answer : They are local variables, accessible only within the `Multi()` method.

5. Why use `Integer` instead of `Double`?

- Answer : `Integer` is appropriate for whole numbers. Use `Double` for decimal values.

Key Salesforce-Specific Concepts

1. Apex Execution Context :

- Code runs in a transactional context on Salesforce servers. Static methods are often used in triggers/batch jobs.

2. Debugging :

- `System.debug()` is critical for troubleshooting in Salesforce, where direct console output isn't available.

3. Testing :

- Salesforce requires 75% test coverage for deployment. Tests are written in Apex and annotated with `@isTest`.

4. Best Practices :

- Follow naming conventions (e.g., `multi` instead of `Multi` for method names).

- Avoid hardcoding values (use parameters or custom settings).

How to Run This Code

1. Execute in Developer Console :

- Open Developer Console > Debug > Open Execute Anonymous Window .

- Run:

```apex
firstClass1.Multi(); // Outputs 20 in debug logs.
```

2. Deployment :

- Deploy the class to a Salesforce org only after writing a test class (required by Salesforce).

Common Pitfalls

- Hardcoded Values : The method always multiplies 4 and 5. Make it reusable with parameters.
- Naming Conventions : Use camelCase for method names (e.g., `multi` instead of `Multi`).
- Lack of Error Handling : Add validation for null/zero values if inputs are dynamic.

Data Types in Apex

Apex supports both primitive and composite data types. Here's a breakdown:

1. Primitive Data Types

These are basic types built into the language:

Data Type	Description & Example
Integer	Whole numbers (e.g., Integer num = 10;).
Double	Decimal numbers (e.g., Double price = 19.99;).
Long	Large integers (e.g., Long bigNumber = 2147483648L;).
String	Text (e.g., String name = 'Salesforce';).
Boolean	true or false (e.g., Boolean isActive = true;).
Date	Date without time (e.g., Date today = Date.today();).
Datetime	Date and time (e.g., Datetime now = Datetime.now();).
Time	Time without a date (e.g., Time meetingTime = Time.newInstance(14, 30, 0, 0);).
ID	Salesforce record ID (e.g., Id accountId = '001xx000003DGb0';).
Blob	Binary data (e.g., Blob fileData = Blob.valueOf('Hello World');).
Object	Generic type (e.g., Object obj = 10; // Can hold any data type).