**CDAC Mumbai LabAssignment**

# Section1:Error-DrivenLearninginJava

**Objective:**Thisassignmentfocusesonunderstandingandfixingcommonerrorsencounteredin Java programming. Byanalyzing and correcting the provided code snippets, you will develop a deeper understanding of Java's syntax, data types, and control structures.

**Instructions:**

1. **IdentifytheErrors:**Revieweachcodesnippettoidentifytheerrorsorissues present.
2. **ExplaintheError:** Writeabriefexplanationoftheerroranditscause.
3. **FixtheError:**Modifythecodetocorrecttheerrors.Ensurethatthecodecompiles andrunsas expected.
4. **SubmitYourWork:**Providethecorrectedcodealongwithexplanationsforeachsnippet.

#### Snippet1:

publicclassMain{

public void main(String[] args) { System.out.println("Hello,World!");

}

}

### Whaterrordoyougetwhenrunningthiscode?

### error: // static are missing

### Solution of Snippet 1:

publicclassMain{

public static void main(String[] args) { System.out.println("Hello,World!");

}

}

#### Snippet2:

publicclassMain{

static void main(String[] args) { System.out.println("Hello,World!");

}

}

### Whathappenswhenyoucompileandrunthiscode?

### -error: // public are missing // main method not found

### Solution of Snippet 2:

publicclassMain{

public static void main(String[] args) { System.out.println("Hello,World!");}}

#### Snippet3:

publicclassMain{

public static int main(String[] args) { System.out.println("Hello,World!"); return 0;

}

### What error do you encounter? Why is void used in the main method?

* **error:** // main method not found -> public static void main

### Solution of Snippet 3:

publicclassMain{

public static void main(String[] args) { System.out.println("Hello,World!"); return 0;

#### Snippet4:

publicclassMain{

A close up of a logo

Description automatically generatedpublic static void main() { System.out.println("Hello,World!");

}

}

### error: // argument and string not declare

### Solution

public class Main {

public static void main(String[] args) {

System.out.println("Hello, World!");

}

}

#### Snippet5:

publicclassMain{

public static void main(String[] args) { System.out.println("MainmethodwithString[]args");

}

public static void main(int[] args) { System.out.println("Overloadedmainmethodwithint[]args");

}

}

### Canyouhavemultiplemainmethods?Whatdoyouobserve?

// yes we use multiple main method - > method overloading

#### Snippet6:

publicclassMain{

publicstaticvoidmain(String[]args){ int x = y + 10;System.out.println(x);

}

}

### What erroroccurs?Whymustvariablesbedeclared?

### // y is not declared. Bcoz of java need to execute program so the datatype required

### Solution

### public class Main {

### public static void main(String[] args) {

### int y = 5; // Declare and initialize 'y'

### int x = y + 10;

### System.out.println(x);

### }

### }

#### Snippet7:

publicclassMain{

publicstaticvoidmain(String[]args){ int x = "Hello"; System.out.println(x);

}

}

### Whatcompilationerrordoyousee?WhydoesJavaenforcetype safety?

### //Compilation Error occur

### Solution

### public class Main {

### public static void main(String[] args) {

### String x = "Hello";

### System.out.println(x);

### }

### }

#### Snippet8:

publicclassMain{

publicstaticvoidmain(String[]args){ System.out.println("Hello, World!"

}

}

### Whatsyntaxerrorsarepresent?Howdotheyaffect compilation?

### //parentheses missing

### Solution

### public class Main {

### public static void main(String[] args) {

### System.out.println("Hello, World!");

### }

### }

#### Snippet9:

publicclassMain{

publicstaticvoidmain(String[]args){ int class = 10; System.out.println(class);

}

}

### Whaterroroccurs?Whycan'treservedkeywordsbeusedasidentifiers?

### error cause:-

### // class is a reserved keyword

### Solution

### public class Main {

### public static void main(String[] args) {

### int number = 10;

### System.out.println(number);

### }

### }

#### Snippet10:

public class Main { publicvoiddisplay(){

System.out.println("Noparameters");

}

public void display(int num) { System.out.println("Withparameter:"+num);

}

publicstaticvoidmain(String[]args){ display();

display(5);

}

}

### Whathappenswhenyoucompileandrunthiscode?Ismethodoverloadingallowed?

### Error: Compilation Error: //non-static method () cannot be referenced from a static

### Solution of Snippet 10:

### public class Main {

### public void display() {

### System.out.println("No parameters");

### }

### public void display(int num) {

### System.out.println("With parameter: " + num);

### }

### public static void main(String[] args) {

### Main obj = new Main(); // Create an instance of Main class

### obj.display(); // Call the instance method display() with no parameters

### obj.display(5); // Call the overloaded method display(int num) with an integer parameter

### }

#### Snippet11:

publicclassMain{

publicstaticvoidmain(String[]args){ int[] arr = {1, 2, 3}; System.out.println(arr[5]);

}

}

### Whatruntimeexceptiondoyouencounter?Whydoesit occur?

### error: When you run this code, you will encounter a java.lang.ArrayIndexOutOfBoundsException.

### - The runtime exception ArrayIndexOutOfBoundsException occurs because the code attempts to access an array index (5) that is out of bounds for the given array (arr with valid indices 0, 1, and 2).

### Solution of Snippet 11:

### public class Main {

### public static void main(String[] args) {

### int[] arr = {1, 2, 3};

### System.out.println(arr[2]); // Accessing a valid index

### }

### }

#### Snippet12:

publicclassMain{

publicstaticvoidmain(String[]args){ while (true) {

System.out.println("InfiniteLoop");

}

}

}

### Whathappenswhenyourunthiscode?Howcanyouavoidinfinite loops?

### effect: // infinite loop occur

### In the below example, the loop runs only 5 times before count < 5 becomes false, and the loop terminates.

### Solution of Snippet 12:

### public class Main {

### public static void main(String[] args) {

### int count = 0;

### while (count < 5) {

### System.out.println("Loop iteration: " + count);

### count++;

### }

### }

### }

#### Snippet13:

publicclassMain{

publicstaticvoidmain(String[]args){ String str = null; System.out.println(str.length());

}

}

### Whatexceptionisthrown?Whydoesitoccur?

### Exception thrown: java.lang.NullPointerException is thrown when you run this code.

* In this code, the variable str is declared and explicitly set to null.
* The line System.out.println(str.length()); attempts to call the length() method on the str object.
* However, since str is null and does not reference any actual String object, there is no object to call the length() method on.
* This results in the NullPointerException being thrown at runtime

### Solution of Snippet 13:

### public class Main {

### public static void main(String[] args) {

### String str = null;

### if (str != null) {

### System.out.println(str.length());

### } else {

### System.out.println("String is null, cannot determine length.");

### }

### }

### }

#### Snippet14:

publicclassMain{

publicstaticvoidmain(String[]args){ double num = "Hello"; System.out.println(num);

}

}

### Whatcompilationerroroccurs?WhydoesJavaenforcedatatypeconstraints?

### error: incompatible types: String cannot be converted to double

### double num = "Hello";

* Java enforces data type constraints to ensure type safety, allowing the compiler to catch type-related errors early and making the code more predictable and easier to maintain.

### Solution of Snippet 14:

### public class Main {

### public static void main(String[] args) {

### double num = 10.5; // Assigning a double value

### System.out.println(num);

### }

### }

### ^

#### Snippet15:

publicclassMain{

publicstaticvoidmain(String[]args){ int num1 = 10;

doublenum2= 5.5;

intresult=num1+num2; System.out.println(result);

}

}

### Whaterroroccurswhencompilingthiscode?Howshouldyouhandledifferentdatatypes in operations?

-**error**: incompatible types: possible lossy conversion from double to int int result = num1 + num2;

- The code will not compile because of an attempt to assign a double result to an int variable, which is not allowed without explicit casting.

-To handle different data types in operations, use explicit type casting if necessary, or choose a variable type that matches the result of the operation to avoid loss of data and ensure correct program behaviour

### Solution of Snippet 15:

public class Main {

public static void main(String[] args) {

int num1 = 10;

double num2 = 5.5;

int result = (int) (num1 + num2); // Explicitly cast the result to int

System.out.println(result);

}

}

#### Snippet16:

publicclassMain{

publicstaticvoidmain(String[]args){ int num = 10;

double result = num / 4; System.out.println(result);

}

}

### Whatistheresultof thisoperation?Istheoutputwhatyouexpected?

### The resulut of this operation is 2.0

### The output is 2.0, which might not be what you expected if you anticipated the operation to consider the fractional part (like 2.5). To achieve 2.5, you would need to make sure that one of the operands is a double or float

### Solution of Snippet 16:

publicclassMain{

publicstaticvoidmain(String[]args){ int num = 10;

double result = (double)num / 4; System.out.println(result);

}

A close up of a logo

Description automatically generated}

### This approach will yield 2.5 as the output.

#### Snippet17:

publicclassMain{

publicstaticvoidmain(String[]args){ int a = 10;

intb=5;

int result = a \*\* b; System.out.println(result);

}

}

### Whatcompilationerroroccurs?Whyisthe\*\*operatornotvalidinJava?

### Error: illegal start of expression

* **Invalid Operator (\*\*)**: In Java, the \*\* operator is not recognized or supported for exponentiation (raising a number to a power). Java does not have a built-in operator for exponentiation like some other languages (e.g., Python). The \*\* operator is not valid in Java, which is why the compiler throws an error.
* **Exponentiation in Java**: To perform exponentiation in Java, you need to use the Math.pow() method. This method takes two arguments, the base and the exponent, and returns the result as a double.

### Solution of Snippet 17:

### public static void main(String[] args) {

### int a = 10;

### int b = 5;

### double result = Math.pow(a, b); // Use Math.pow for exponentiation

### System.out.println(result);

### }

### int result = (int) Math.pow(a, b);

### System.out.println(result);

#### Snippet18:

publicclassMain{

publicstaticvoidmain(String[]args){ int a = 10;

intb=5;

int result = a + b \* 2; System.out.println(result);

}

}

### Whatistheoutputofthiscode?Howdoesoperatorprecedenceaffecttheresult?

### The result of the expression a + b \* 2 is 20, which is what the code will output.

### Operator Precedence: In Java (and many other programming languages), operators have a specific order of precedence, which dictates the order in which operations are performed.

### Multiplication (\*) Before Addition (+): Multiplication has higher precedence than addition. Therefore, in the expression a + b \* 2, the multiplication b \* 2 is performed first, before the addition.

### Step-by-Step Evaluation:

### b \* 2 is evaluated first: 5 \* 2 = 10.

### Then, a + 10 is evaluated: 10 + 10 = 20.

#### Snippet19:

publicclassMain{

publicstaticvoidmain(String[]args){ int a = 10;

intb=0;

int result = a / b; System.out.println(result);

}

}

### Whatruntimeexceptionisthrown?WhydoesdivisionbyzerocauseanissueinJava?

Exception in thread "main" java.lang.ArithmeticException: / by zero

Issue for cause:

**Mathematical Reason**: In mathematics, division by zero is undefined because there is no number that, when multiplied by zero, gives a non-zero number. This is why any attempt to divide by zero leads to an error.

**Java's Handling**: Java throws an ArithmeticException at runtime to prevent the program from continuing with an undefined or invalid operation

### Solution of Snippet 19:To avoid this exception, you should always check if the divisor is zero before performing the division.

### if (b != 0) {

### int result = a / b;

### System.out.println(result);

### } else {

### System.out.println("Cannot divide by zero.");

### }

#### Snippet20:

publicclassMain{

publicstaticvoidmain(String[]args){ System.out.println("Hello, World")

}

}

### Whatsyntaxerroroccurs?Howdoesthemissingsemicolonaffectcompilation?

### Explanation:

* **Missing Semicolon**: In Java, every statement must end with a semicolon (;). The semicolon acts as a statement terminator, telling the compiler where one statement ends and the next begins.
* **Effect on Compilation**: Without the semicolon, the Java compiler does not recognize the end of the statement, leading to a syntax error. The compiler expects a semicolon at the end of the System.out.println("Hello, World") statement. Because of this error, the program will not compile, and you'll need to fix the code before it can run.

#### Snippet21:

publicclassMain{

publicstaticvoid main(String[] args) { System.out.println("Hello,World!");

//Missingclosingbrace here

}

### Whatdoesthecompilersayaboutmismatchedbraces?

### The compiler will typically produce an error like:

### error: reached end of file while parsing.

#### Snippet22:

publicclassMain{

publicstaticvoidmain(String[]args){ static void displayMessage() {

System.out.println("Message");

}

}

}

### Whatsyntaxerroroccurs?Canamethodbedeclaredinsideanothermethod?

### error: illegal start of expression

### static void displayMessage() {

* **Method Declaration Rules**: In Java, methods cannot be declared inside other methods. All methods must be declared at the class level, directly inside the class body, but not inside other methods or constructors.
* **Why This is an Error**: The main method is trying to declare a new method (displayMessage) within its body, which violates the rules of method declaration in Java. The keyword static and the method signature are not allowed within the scope of another method.

### Solution of Snippet 22:

### public class Main {

### public static void main(String[] args) {

### displayMessage(); // Call the method

### }

### static void displayMessage() {

### System.out.println("Message");

### }

### }

#### Snippet23:

publicclassConfusion{

publicstaticvoidmain(String[]args){ int value = 2;

switch(value){ case 1:

System.out.println("Valueis1");

case 2:

System.out.println("Valueis2");

case 3:

System.out.println("Valueis3"); default:

System.out.println("Defaultcase");

}

}

}

•**Error to Investigate**: Why does the default case print after "Value is 2"? How can you prevent the program from executing the default case?

* **Explanation:**
* **Switch Case Fall-Through**: In Java, once a matching case is found in a switch statement, all the subsequent cases are executed until a break statement or the end of the switch block is encountered. This is known as "fall-through."
* **Why Default Case Prints**: Since there is no break statement after case 2, the program continues to execute case 3 and then the default case, printing all of them.
* **How to Prevent Fall-Through:**
* To prevent the program from executing subsequent cases after a match, you should add break statements at the end of each case:

**Solution of Snippet** 23**:**

public class Confusion {

public static void main(String[] args) {

int value = 2;

switch(value) {

case 1:

System.out.println("Value is 1");

break;

case 2:

System.out.println("Value is 2");

break;

case 3:

System.out.println("Value is 3");

break;

default:

System.out.println("Default case");

break;

}

}

}

#### Snippet24:

publicclassMissingBreakCase{

publicstaticvoidmain(String[]args){ int level = 1;

switch(level){ case 1:

System.out.println("Level1");

A close up of a logo

Description automatically generatedcase 2:

System.out.println("Level2");

case 3:

System.out.println("Level3"); default:

System.out.println("Unknownlevel");

}

}

}

•**Error to Investigate**: When level is 1, why does it print "Level 1", "Level 2", "Level 3", and "Unknown level"? What is the role of the break statement in this situation?

* **Switch Case Fall-Through**: In Java, when a switch statement executes, it starts at the matching case and continues executing subsequent cases until it encounters a break statement or reaches the end of the switch block. This is known as "fall-through."

### Role of the break Statement:

* **Preventing Fall-Through**: The break statement is used to terminate the switch case after the matching case has been executed. When the break statement is encountered, the program exits the switch block, preventing any further cases from being executed.
* **Correct Usage**: To stop the fall-through and ensure that only the matching case is executed, you should add break statements at the end of each case.

**Solution of Snippet** 24**:**

public class MissingBreakCase {

public static void main(String[] args) {

int level = 1;

switch(level) {

case 1:

System.out.println("Level 1");

break;

case 2:

System.out.println("Level 2");

break;

case 3:

System.out.println("Level 3");

break;

default:

System.out.println("Unknown level");

break;

}

}

}

#### Snippet25:

publicclassSwitch{

publicstaticvoidmain(String[]args){ double score = 85.0;

switch(score){ case 100:

System.out.println("Perfectscore!"); break;

case 85:

System.out.println("Greatjob!"); break;

default:

System.out.println("Keeptrying!");

}

}

}

•**Error to Investigate**: Why does this code not compile? What does the error tell you about the types allowed in switch expressions? How can you modify the code to make it work?

**Ans:** The code does not compile because the switch statement in Java does not support double(orfloat)values.Theswitchstatementcanonlybeusedwiththefollowingtypes: byte, short, int, char, enum, String and var.

-To make the code work, you can either convert the double value to an int, use a different type supported by switch, or use an if-else structure instead.

**Solution of Snippet** 25**:**

public class Switch {

public static void main(String[] args) {

int score = 85; // Change to int

switch(score) {

case 100:

System.out.println("Perfect score!");

break;

case 85:

System.out.println("Great job!");

break;

default:

System.out.println("Keep trying!");

}

}

}

#### Snippet26:

publicclassSwitch{

publicstaticvoidmain(String[]args){ int number = 5;

switch(number){ case 5:

System.out.println("Numberis5");

break; case 5:

System.out.println("Thisisanothercase5"); break;

default:

System.out.println("Thisisthedefaultcase");

}

}

}

•**Error to Investigate:** Why does the compiler complain about duplicate case labels? What happens when you have two identical case labels in the same switch block?

**Ans:**Thecompilercomplainaboutduplicatecaselabelsbecause:

InJava,withinaswitchstatement,eachcaselabelmustbeuniquewithintheswitchblock. The compiler complains about duplicate case labels because each case label must be distinct; otherwise, the compiler cannot determine which block of code to execute.

When youhave twoidentical caselabelsin the sameswitch block inJava,thecompilerwill generate an error. This is because having duplicate case labels creates ambiguity in which blockofcodeshouldbeexecuted,makingitimpossibleforthecompilertodeterminewhich code should run when the switch statement matches that value.

**Solution of Snippet** 26**:**

publicclassSwitch{

publicstaticvoidmain(String[]args){ int number = 5;

switch(number){ case 5:

System.out.println("Numberis5");

//Noneedforanothercase5 break;

default:

System.out.println("Thisisthedefaultcase");

}

}

}

# Section2:JavaProgrammingwithConditionalStatements

## Question1:GradeClassification

Writeaprogramtoclassifystudentgradesbasedonthefollowingcriteria:

* + Ifthescoreisgreaterthanorequalto90,print"A"
  + Ifthescoreisbetween80and89,print"B"
  + Ifthescoreisbetween70and79,print"C"
  + Ifthescoreisbetween60and69,print"D"
  + Ifthescoreislessthan60,print"F"

**Ans:**

**classGrade{**

**publicstaticvoidmain(Stringargs[]){ int marks = 50;**

**if(marks>=90){ System.out.println("GradeA");**

**}elseif(marks>=80&&marks<=89){ System.out.println("Grade B");**

**}elseif(marks>=70&&marks<=79){ System.out.println("Grade C");**

**}elseif(marks>=60&&marks<=69){ System.out.println("Grade D");**

**}else if(marks<60){ System.out.println("GradeF");**

**}**

**}**

**}**

## Question2:DaysoftheWeek

Writeaprogramthatusesanestedswitchstatementtoprintout thedayoftheweekbasedonan integerinput(1forMonday,2forTuesday,etc.).Additionally,withineachday,printwhetherit is a weekday or weekend.

**Ans:**

**classDays{**

**publicstaticvoidmain(Stringargs[]){ int day = 4;**

**switch(day){ case 1 :**

**System.out.println("TodayisMonday"); break;**

**case2:**

**System.out.println("Today isTuesday"); break;**

**case3:**

**System.out.println("Today is Wednesday"); break;**

**case4:**

A close up of a logo

Description automatically generated**System.out.println("Today is Thursday") break;**

**case5:**

**System.out.println("Today is Friday"); break;**

**case6:**

**System.out.println("Today is Saturday"); break;**

**case7:**

**System.out.println("Today is Sunday"); break;**

**case 8 : System.out.println("InvalidDay"); break;**

**}**

**}**

**}**

## Question3: Calculator

Writeaprogramthatactsasasimplecalculator.Itshouldaccepttwonumbersandanoperator (+, -, \*, /) asinput. Usea switch statement to perform theappropriate operation. Usenested if- else to check if division by zero is attempted and display an error message.

**Ans:**

**importjava.util.Scanner; class Calculator{**

**publicstaticvoidmain(Stringargs[]){ Scanner sc = new Scanner(System.in);**

**doublenum1,num2,result; char operator;**

**|System.out.println("Enterfirstnumber:"); num1 = sc.nextDouble(); System.out.println("Enter Operator : "); operator = sc.next().charAt(0);**

**System.out.println("Entersecondnumber:"); num2 = sc.nextDouble();**

**if(operator == '+'){ result=num1+num2;**

**}elseif(operator=='-'){ result = num1 - num2;**

**}elseif(operator=='\*'){ result = num1 \* num2;**

**}elseif(operator=='/'){ if(num2 != 0){**

A close up of a logo

Description automatically generated**result=num1/num2;**

**}else{**

**System.out.println("Error:Divisionbyzeronotallowed"); return;**

**}**

**}else{**

**System.out.println("InvalidOperator"); return;**

**}**

**System.out.println("Result:"+result);**

**}**

**}**

## Question4:DiscountCalculation

Writeaprogramtocalculatethediscountbasedonthetotalpurchaseamount.Usethefollowing criteria:

* + IfthetotalpurchaseisgreaterthanorequaltoRs.1000,applya20%discount.
  + IfthetotalpurchaseisbetweenRs.500andRs.999,applya10%discount.
  + IfthetotalpurchaseislessthanRs.500,applya5%discount.

Additionally,iftheuserhasamembershipcard,increasethediscountby5%.

**Ans:**

**classDiscount{**

**publicstaticvoidmain(Stringargs[]){ int price = 200;**

**doubletotal;**

**booleanmembership=false; if(price>=1000){ if(membership == true){ total = price \* 0.25;**

**System.out.println("Total:"+total);**

**}else{**

**total=price\* 0.2;**

**System.out.println("Total:"+total);**

**}**

**}elseif(price>=500&&price<=999){ if(membership == true){**

**total = price \* 0.15;| System.out.println("Total:"+total);**

**}else{**

**total = price \* 0.1; System.out.println("Total:"+total);**

**}**

A close up of a logo

Description automatically generated**}else if(price<500){ if(membership==true){ total = price \* 0.1;**

**System.out.println("Total:"+total);**

**}else{**

**total = price \* 0.05; System.out.println("Total:"+total);**

**}**

**}**

**}**

**}**

## Question5:StudentPass/FailStatuswithNested Switch

Write a program that determines whether a student passes or fails based on their grades in three subjects.Ifthestudentscoresmorethan40inallsubjects,theypass.Ifthestudentfailsinoneor more subjects, print the number of subjects they failed in.

**Ans:**

**publicclassStudentPassFail{**

**publicstaticvoidmain(String[]args){**

**// Example grades for three subjects int grade1 =45;//Scorefor Subject 1 intgrade2 =38;// Scorefor Subject 2 intgrade3=50;//ScoreforSubject3**

**//Initializethecountoffailedsubjects int failedCount = 0;**

**//Nestedswitchcasetocheckgradesforeachsubject for (int subject = 1; subject <= 3; subject++) {**

**intgrade;**

**//Determinewhichsubject'sgradetouse switch (subject) {**

**case 1:**

**grade=grade1; break;**

**case2:**

**grade=grade2; break;**

**case 3:**

**grade=grade3; break;**

**default:**

A close up of a logo

Description automatically generated**grade=0;//Defaultcase,thoughthisshouldnotoccur**

**}**

**//Nestedswitchcasetodetermineifthegradeispassingorfailing switch (grade) {**

**case0://Specialcaseifgradesareoutsideexpectedrange(forsafety) case 1:**

**case 2:**

**case 3:**

**case 4:**

**case 5:**

**case 6:**

**case 7:**

**case 8:**

**case 9:**

**case 10:**

**case 11:**

**case 12:**

**case 13:**

**case 14:**

**case 15:**

**case 16:**

**case 17:**

**case 18:**

**case 19:**

**case 20:**

**case 21:**

**case 22:**

**case 23:**

**case 24:**

**case 25:**

**case 26:**

**case 27:**

**case 28:**

**case 29:**

**case 30:**

**case 31:**

**case 32:**

**case 33:**

A close up of a logo

Description automatically generated

**case 34:**

**case 35:**

**case 36:**

**case 37:**

**case 38:**

**case 39:**

**case 40:**

**//Ifgradeis40orless,considerasfail failedCount++;**

**break; default:**

**//Ifgradeismorethan40,consideraspass break;**

**}**

**}**

**//Determineandprinttheresultbasedonthecountoffailedsubjects if (failedCount == 0) {**

**System.out.println("Thestudentpasses.");**

**}else{**

**System.out.println("Thestudentfailedin"+failedCount+"subject(s).");**

**}**

**}**

**}**

# Section3:FoodforThought:ResearchandReadMore About

#### EvolutionofProgrammingLanguages

* + **ResearchTopic:**Explorethedifferentlevelsofprogramminglanguages:Low-level,High-level, and Assembly-level languages.

### QuestionstoPonder:

* + - * WhatisaLow-levellanguage?Giveexamplesandexplainhowtheywork.
      * WhatisaHigh-levellanguage?Howdoesitdifferfromalow-levellanguagein terms of abstraction and usage?
      * WhatisanAssembly-level language,andwhatroledoesit playin programming?
      * Whydoweneeddifferentlevelsofprogramminglanguages?Whatarethetrade- offs between simplicity and control over the hardware?

#### DifferentProgrammingLanguagesandTheirUsage

* + **ResearchTopic:**Exploredifferentprogramminglanguagesandunderstandtheiruse cases.

### QuestionstoPonder:

* + - * WhatarethestrengthsandweaknessesoflanguageslikeC,Python,Java, JavaScript, C++, Ruby, Go, etc.?
      * In which scenarios would you choose a specific language over others? For example,whywouldyouuseJavaScriptforwebdevelopmentbutPythonfor data science?
      * Canoneprogramminglanguagebeusedforalltypesofsoftwaredevelopment? Why or why not?

#### WhichProgrammingLanguageistheBest?

* + **ResearchTopic:**Investigatethedebatearoundthe"best"programminglanguage.

### QuestionstoPonder:

* + - * Istheretrulya"best" programminglanguage?Ifso,whichone,andwhy?
      * If a language is consideredthe best,whyaren’t all organizations usingit? What factorsinfluencethechoiceofaprogramminglanguageinanorganization(e.g., cost, performance, ecosystem, or community support)?
      * Howdotrendsinprogramminglanguagesshiftovertime?Whataresome emerging languages, and why are they gaining popularity?

#### FeaturesofJava

* + **ResearchTopic:** DivedeepintothefeaturesofJava.

### QuestionstoPonder:

* + - * WhyisJavaconsideredplatform-independent?HowdoestheJVMcontributeto this feature?
      * WhatmakesJavarobust?Considerfeatureslikememorymanagement,exception handling, and type safety. How do these features contribute to its robustness?
      * WhyisJavaconsideredsecure?Explorefeatureslikebytecodeverification, automatic garbage collection, and built-in security mechanisms.
      * Analyzeotherfeatureslikemultithreading,portability,andsimplicity.Whyare they important, and how do they impact Java development?

#### Roleofpublicstaticvoidmain(String[]args)(PSVM)

* + **ResearchTopic:**AnalyzethestructureandpurposeofthemainmethodinJava.

### QuestionstoPonder:

* + - * Whatistheroleofeachkeywordin publicstaticvoid main(String[] args)?
      * What would happen if one of these keywords (public, static, or void) were removedoraltered?Experimentbymodifyingthemainmethodandnotedown the errors.
      * WhyistheString[]args parameterusedinthemainmethod?Whatdoesitdo,and what happens if you omit it?

#### CanWeWriteMultiplemainMethods?

* + **ResearchTopic:** Experimentwithmultiple mainmethodsinJava.

### QuestionstoPonder:

* + - * Canaclasshavemorethanone main method?Whatwouldhappenifyoutriedto define multiple main methods in a single class?
      * Whathappensifmultipleclassesinthesameproject havetheirownmain

methods?Howdoes theJavacompilerandJVMhandlethis situation?

* + - * Investigatemethodoverloadingforthemainmethod.Canyouoverloadthemain

methodwithdifferentparameters,andhowdoesthisaffectprogramexecution?

#### NamingConventionsinJava

* + **ResearchTopic:**InvestigateJava'snamingconventions.

### QuestionstoPonder:

* + - * WhydosomewordsinJavastartwithuppercase(e.g.,Classnames)whileothers are lowercase (e.g., variable names and method names)?
      * Whataretherulesfornamingvariables,classes,andmethodsinJava,andwhyis following these conventions important?
      * Howdonamingconventionsimprovecodereadabilityandmaintainability, especially in large projects?

#### JavaObjectCreationandMemoryManagement

* + **ResearchTopic:** UnderstandJava’sapproachtoobjectsand memory.

### QuestionstoPonder:

* + - * WhyareJavaobjectscreatedontheheap,andwhataretheimplications ofthis?
      * HowdoesJavamanagememory,andwhatroledoesthegarbagecollectorplay?
      * Whatarethedifferencesbetweenmethodoverloadingandmethodoverridingin Java?
      * WhatistheroleofclassesandobjectsinJava?Explorehowtheysupportthe principles of object-oriented programming (OOP), such as encapsulation, inheritance, and polymorphism.

#### PurposeofAccessModifiersinJava

* + **ResearchTopic:**Explorethepurposeofaccessmodifiersin Java.

### QuestionstoPonder:

* + - * Whatisthepurposeofaccessmodifiers(e.g.,public,private)incontrolling access to classes, methods, and variables?
      * Howdoaccessmodifierscontributetoencapsulation,dataprotection,and security in object-oriented programming?
      * Howdoaccessmodifiersinfluencesoftwaredesignand maintenance?
  + Considerpotentialchallengesorlimitationsofautomaticmemorymanagement.