**Lab Assignment 2**

**Section 1: Error-Driven Learning in Java**

Snippet 1:

public class Main {

public void main(String[] args) {

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

} }

• What error do you get when running this code?

Error: Main method is not static in class Main, please define the main method as:

public static void main(String[] args) (Runtime Error)

**public class Main {**

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

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

**}**

**}**

Snippet 2:

public class Main {

static void main(String[] args) {

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

} }

• What happens when you compile and run this code?

Error: Main method not found in class Main, please define the main method as:

public static void main(String[] args)

or a JavaFX application class must extend javafx.application.Application

**public class Main {**

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

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

**} }**

Snippet 3:

public class Main{

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 must return a value of type void in class Main, please

define the main method as: public static void main(String[] args)

**public class Main {**

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

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

**} }**

**void:** It is a **keyword**and is used to **specify that a method doesn’t return anything.**

Snippet 4:

public class Main {

public static void main() {

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

} }

• What happens when you compile and run this code? Why is String[] args needed?

Error: Main method not found in class Main, please define the main method as:

public static void main(String[] args)

or a JavaFX application class must extend javafx.application.Application

**public class Main {**

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

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

**} }**

Why is String[] args needed?

**String[] args is needed to allow a Java program to accept and use command-line arguments, providing a way to interact with the program when it is executed.**

Snippet 5:

public class Main {

public static void main(String[] args) {

System.out.println("Main method with String[] args"); }

public static void main(int[] args) {

System.out.println("Overloaded main method with int[] args");

} }

**O/P: Main method with String[] args**

• Can you have multiple main methods? What do you observe?

Yes we can have multiple main methods. But public static void main(String args[]) is giving output only after compiling and running.

Snippet 6:

public class Main {

public static void main(String[] args) {

int x = y + 10;

System.out.println(x);

} }

• What error occurs? Why must variables be declared?

(Compile time)error: cannot find symbol

int x = y + 10;

^

symbol: variable y

location: class Main

**public class Main {**

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

**int y=0;**

**int x = y + 10;**

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

**} }**

**Why must variables be declared?**

**because Java is a statically-typed language, which means that the data type of a variable must be declared before it is used.**

Snippet 7:

public class Main {

public static void main(String[] args) {

int x = "Hello";

System.out.println(x);

} }

error: incompatible types: String cannot be converted to int

int x = "Hello";

^

**public class Main {**

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

**String x = "Hello";**

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

**} }**

• What compilation error do you see? Why does Java enforce type safety?

**Java prevents the programs from accessing memory in inappropriate ways by controlling the memory access of each object.**

Snippet 8:

public class Main {

public static void main(String[] args) {

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

} }

**public class Main {**

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

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

**} }**

• What syntax errors are present? How do they affect compilation?

error: ')' expected

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

How do they affect compilation?

Because of this error the rest of the code might not even be checked for further errors until the initial syntax error is corrected.

Snippet 9:

public class Main {

public static void main(String[] args) {

int class = 10;

System.out.println(class);

} }

**public class Main {**

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

**int num = 10;**

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

**} }**

• What error occurs? Why can't reserved keywords be used as identifiers?

Main.java:3: error: not a statement

int class = 10;

^

Main.java:3: error: ';' expected

int class = 10;

^

Main.java:3: error: <identifier> expected

int class = 10;

^

Main.java:4: error: illegal start of expression

System.out.println(class);

^

Main.java:4: error: <identifier> expected

System.out.println(class);

Why can't reserved keywords be used as identifiers?

Reserved words have predefined meanings in the language's syntax and using them as identifiers would lead to confusion and syntax errors.

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) {

display();

display(5);

} }

**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 the Main class**

**obj.display(); // Call the method on the instance**

**obj.display(5); // Call the overloaded method on the instance**

**} }**

• What happens when you compile and run this code? Is method overloading allowed?

Main.java:9: error: non-static method display() cannot be referenced from a static context

display();

^

Main.java:10: error: non-static method display(int) cannot be referenced from a static context

display(5);

^

the main method is trying to call the display() method directly without specifying an object. In Java, non-static methods (like display() methods) cannot be called directly from a static context (like the main method) unless they are called on an instance of the class.

**Is method overloading allowed?**

**Yes method overloading is allowed.**

Snippet 11:

public class Main {

public static void main(String[] args) {

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

} }

• What runtime exception do you encounter? Why does it occur?

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length at Main.main(Main.java:4)

**Why does it occur?**

**occurs because an index of an array that is outside its valid range. In Java, this exception typically indicates that you are attempting to access an array element at an index that does not exist.**

Snippet 12:

public class Main {

public static void main(String[] args) {

while (true) {

System.out.println("Infinite Loop");

} } }

**public class Main {**

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

**while (true) {**

**System.out.println("Infinite Loop");**

**break;**

**} } }**

• What happens when you run this code? How can you avoid infinite loops?

It is running continuously.

How can you avoid infinite loops?

By giving proper termination conditions.

Snippet 13:

public class Main {

public static void main(String[] args) {

String str = null;

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

} }

**public class Main {**

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

**String str = "Cdac Kharghar";**

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

**} }**

• What exception is thrown? Why does it occur?

Exception in thread "main" java.lang.NullPointerException: Cannot invoke "String.length()" because "<local1>" is null

at Main.main(Main.java:4)

**Why does it occur?**

**trying to invoke the length() method on a String object that is null.**

Snippet 14:

public class Main {

public static void main(String[] args) {

double num = "Hello";

System.out.println(num);

} }

**public class Main {**

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

**String num = "Hello";**

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

**} }**

• What compilation error occurs? Why does Java enforce data type constraints?

Main.java:3: error: incompatible types: String cannot be converted to double

double num = "Hello";

**Why does Java enforce data type constraints?**

**to improve code readability and catch type-related errors at compile time.**

Snippet 15:

public class Main {

public static void main(String[] args) {

int num1 = 10;

double num2 = 5.5;

int result = num1 + num2;

System.out.println(result);

} }

**public class Main {**

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

**int num1 = 10;**

**double num2 = 5.5;**

**int result = num1 +(int) num2;**

**System.out.println(result); } }**

• What error occurs when compiling this code? How should you handle different data types in operations?

Main.java:5: error: incompatible types: possible lossy conversion from double to int

int result = num1 + num2;

**How should you handle different data types in operations?**

**By explicit type casting.**

Snippet 16:

public class Main {

public static void main(String[] args) {

int num = 10;

double result = num / 4;

System.out.println(result);

} }

• What is the result of this operation? Is the output what you expected?

2.0

**Is the output what you expected?**

**Yes**

Snippet 17:

public class Main {

public static void main(String[] args) {

int a = 10;

int b = 5;

int result = a \*\* b;

System.out.println(result);

} }

• What compilation error occurs? Why is the \*\* operator not valid in Java?

Main.java:5: error: illegal start of expression

int result = a \*\* b;

**Why is the \*\* operator not valid in Java?**

**In Java, the \*\* operator is not valid because exponentiation is handled using the Math.pow() method.**

Snippet 18:

public class Main {

public static void main(String[] args) {

int a = 10;

int b = 5;

int result = a + b \* 2;

System.out.println(result);

} }

• What is the output of this code? How does operator precedence affect the result?

O/P : 20

**How does operator precedence affect the result?**

**Operator precedence affects the order of evaluation in expressions. Operators with higher precedence are evaluated before those with lower precedence.**

Snippet 19:

public class Main {

public static void main(String[] args) {

int a = 10; int b = 0;

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

} }

• What runtime exception is thrown? Why does division by zero cause an issue in Java?

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

at Main.main(Main.java:4)

**Why does division by zero cause an issue in Java?**

**because it leads to undefined or mathematically invalid operations.**

Snippet 20:

public class Main {

public static void main(String[] args) {

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

} }

**public class Main {**

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

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

**} }**

• What syntax error occurs? How does the missing semicolon affect compilation?

Main.java:3: error: ';' expected

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

**How does the missing semicolon affect compilation?**

**A missing semicolon in Java leads to compilation errors because it disrupts the syntax and structure of the code.**

Snippet 21:

public class Main {

public static void main(String[] args) {

System.out.println("Hello, World!"); // Missing closing brace here }

• What does the compiler say about mismatched braces?

Main.java:3: error: reached end of file while parsing

System.out.println("Hello, World!"); // Missing closing brace here }

Snippet 22:

public class Main {

public static void main(String[] args) {

static void displayMessage() {

System.out.println("Message");

} } }

• What syntax error occurs? Can a method be declared inside another method?

Main.java:3: error: illegal start of expression

static void displayMessage() {

^

Main.java:5: error: class, interface, enum, or record expected

} } }

**Can a method be declared inside another method?**

**Java does not support “directly” nested methods.**

Snippet 23:

public class Confusion {

public static void main(String[] args) {

int value = 2;

switch(value) {

case 1:

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

case 2:

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

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

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

} } }

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

the default case print after "Value is 2" because there is no break statement after case 2.

**How can you prevent the program from executing the default case?**

**To prevent the default case from executing, use break statements at the end of each case block. This ensures that once a case block is executed, the control flow exits the switch statement, avoiding any fall-through to the default case.**

Snippet 24:

public class MissingBreakCase {

public static void main(String[] args) {

int level = 1;

switch(level) {

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

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

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

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

} } }

• 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?

Because there is no break statement after every case.

the role of the break statement in this situation is it will execute only one case if break uses.

Snippet 25:

public class Switch {

public static void main(String[] args)

{ double score = 85.0; switch(score) {

case 100: System.out.println("Perfect score!");

break;

case 85: System.out.println("Great job!");

break; default: System.out.println("Keep trying!");

} } }

• 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?

error: constant label of type int is not compatible with switch selector type double.

error tell about the types allowed in switch expressions because double can’t be converted to int .

we can take initialize score as int instead of double.

Snippet 26:

public class Switch {

public static void main(String[] args) {

int number = 5;

switch(number) {

case 5: System.out.println("Number is 5");

break;

case 5: System.out.println("This is another case 5");

break;

default: System.out.println("This is the default case");

} } }

**• 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?**

**The Java compiler enforces unique case labels within a switch statement to avoid ambiguity and ensure that each possible value of the switch expression maps to a distinct block of code. This prevents logical errors and ensures that the switch statement functions as intended.**

**Section 2: Java Programming with Conditional Statements**

**Question 1: Grade Classification**

**Write a program to classify student grades based on the following criteria:**

**• If the score is greater than or equal to 90, print "A"**

**• If the score is between 80 and 89, print "B"**

**• If the score is between 70 and 79, print "C"**

**• If the score is between 60 and 69, print "D"**

**• If the score is less than 60, print "F"**

public class Grade

{

public static void main(String args[])

{

int score=88;

if(score>=90)

System.out.println("A");

else if(score>=80 && score<=89)

System.out.println("B");

else if(score>=70 && score<=79)

System.out.println("C");

else if(score>=60 && score<=69)

System.out.println("D");

else

System.out.println("F");

}

}

Question 2: Days of the Week

Write a program that uses a nested switch statement to print out the day of the week based on an integer input (1 for Monday, 2 for Tuesday, etc.). Additionally, within each day, print whether it is a weekday or weekend.

public class Day{

public static void main(String args[])

{

int day=7,input=7;

switch(day)

{

case 1:

System.out.println("Monday");

switch(input)

{

case 1:case 2:case 3:case 4: case 5:

System.out.println("weekday");

}

break;

case 2:

System.out.println("Tuesday");

switch(input)

{

case 1:case 2:case 3:case 4: case 5:

System.out.println("weekday");

}

break;

case 3:

System.out.println("Wednesday");

switch(input)

{

case 1:case 2:case 3:case 4: case 5:

System.out.println("weekday");

}

break;

case 4:

System.out.println("Thursday");

switch(input)

{

case 1:case 2:case 3:case 4: case 5:

System.out.println("weekday");

}

break;

case 5:

System.out.println("Friday");

switch(input)

{

case 1:case 2:case 3:case 4: case 5:

System.out.println("weekday");

}

break;

case 6:

System.out.println("Saturday");

switch(input)

{

case 6:case 7:

System.out.println("weekend");

}

break;

case 7:

System.out.println("Sunday");

switch(input)

{

case 6:case 7:

System.out.println("weekend");

}

break;

default:

System.out.println("Invalid Day");

}

}

}

Question 3: Calculator Write a program that acts as a simple calculator. It should accept two numbers and an operator (+, -, \*, /) as input. Use a switch statement to perform the appropriate operation. Use nested ifelse to check if division by zero is attempted and display an error message.

public class Calculator{

public static void main(String args[])

{

int n1=25,n2=5;

char ch='+';

switch(ch)

{

case '+':

System.out.println(n1+n2);

break;

case '-':

System.out.println(n1-n2);

break;

case '\*':

System.out.println(n1\*n2);

break;

case '/':

if(n2==0)

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

else

System.out.println(n1/n2);

break;

default:

System.out.println("Invalid operator");

}

}

}

Question 4: Discount Calculation

Write a program to calculate the discount based on the total purchase amount. Use the following criteria: • If the total purchase is greater than or equal to Rs.1000, apply a 20% discount. • If the total purchase is between Rs.500 and Rs.999, apply a 10% discount. • If the total purchase is less than Rs.500, apply a 5% discount. Additionally, if the user has a membership card, increase the discount by 5%.

public class Discount{

public static void main(String args[])

{

int rs=500;

double discount;

String membership="no";

if(rs>=1000)

{discount=rs\*20/100;}

else if(rs>=500 && rs<=999)

{discount=rs\*10/100;}

else

{discount=rs\*5/100;}

if(membership=="yes")

{

discount+=rs\*5/100;

System.out.println(discount);

}

System.out.println(discount);

}

}

Question 5: Student Pass/Fail Status with Nested Switch

Write a program that determines whether a student passes or fails based on their grades in three subjects. If the student scores more than 40 in all subjects, they pass. If the student fails in one or more subjects, print the number of subjects they failed in.

public class StdMarks{

public static void main(String args[])

{

int sub1=45,sub2=40,sub3=30;

int cnt=0;

switch (sub1 > 40 ? 1 : 0) {

case 0: // Failed in Subject 1

cnt++;

break;

case 1: // Passed in Subject 1

break;

}

// Nested switch to check Subject 2

switch (sub2 > 40 ? 1 : 0) {

case 0: // Failed in Subject 2

cnt++;

break;

case 1: // Passed in Subject 2

break;

}

// Nested switch to check Subject 3

switch (sub3 > 40 ? 1 : 0) {

case 0: // Failed in Subject 3

cnt++;

break;

case 1: // Passed in Subject 3

break;

}

switch (cnt) {

case 0:

System.out.println("The student passes all subjects.");

break;

default:

System.out.println("The student failed in " + cnt + " subject(s).");

break;

}

}

}