**ASHISH JADHAV\_JUHU**

**ASSIGNMENT 2**

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 not found in class Main, please define the main method as: public static void main(String[] args)

**-------------------------------------------------------------------------------------------------------------------------------------**

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)

**-------------------------------------------------------------------------------------------------------------------------------------**

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)}

^

//Void is used for return type because main method does not return anything.as soon as the main Method terminates the java program also terminates. That’s why Void is used.

**-------------------------------------------------------------------------------------------------------------------------------------**

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: can't find main(String[]) method in class: Main

//String is needed because it stores java command line argument and is an array of type java.lang.string class.

**-------------------------------------------------------------------------------------------------------------------------------------**

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");

}

}

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

//Yes we can use the multiple main method .

//Observation: we can overload the main method by changing the parameter. Main method always start the program with String args. whatever we overload another versions of main method it does not be used as entry point but gets called in code itself.

**-------------------------------------------------------------------------------------------------------------------------------------**

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?

error: cannot find symbol

int x = y + 10;

^

symbol: variable y

location: class Main

//Variables must be declared to state the values of the specific datatype. because it allocates the memory. So to store the memory of the datatype ,variables must be declared.

**-------------------------------------------------------------------------------------------------------------------------------------**

Snippet 7:

public class Main {

public static void main(String[] args) {

int x = "Hello";

System.out.println(x);

}

}

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

error: incompatible types: String cannot be converted to int

int x = "Hello";

^

//Type safety prevents run time error. Enforsing safety does improve security and performance of the program.

**-------------------------------------------------------------------------------------------------------------------------------------**

Snippet 8:

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: ')' or ',' expected

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

// Without completing the round bracket the program wont be compiled .

**-------------------------------------------------------------------------------------------------------------------------------------**

Snippet 9:

public class Main {

public static void main(String[] args) {

int class = 10;

System.out.println(class);

}

}

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

**O/P**

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);

^

5 errors

//Because reserved keywords are integral to the syntax and semantics of the language, using them as identifiers would create ambiguity, making it difficult for the compiler to parse and understand the code.

**-------------------------------------------------------------------------------------------------------------------------------------**

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);

}

}

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

**O/P**

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);

^

2 errors

//MethodOverloading is allowed. the display() method is overloaded.

**-------------------------------------------------------------------------------------------------------------------------------------**

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?

**O/P**

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 3

at Main.main(Main.java:4)

//because the index does not exist in the array that we want to access.

**-------------------------------------------------------------------------------------------------------------------------------------**

Snippet 12:

public class Main {

public static void main(String[] args) {

while (true) {

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

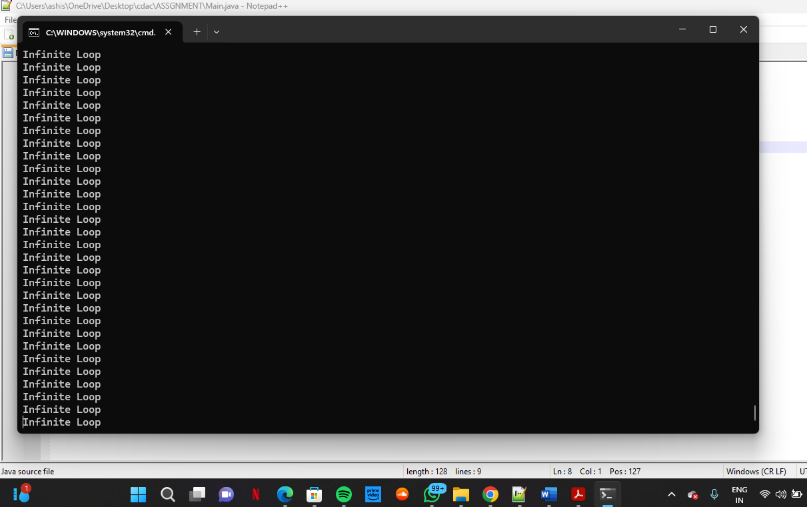
}

}

}

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

**O/P**

****

**//**we can avoid infinite loop by using break or return keyword.

**-------------------------------------------------------------------------------------------------------------------------------------**

Snippet 13:

public class Main {

public static void main(String[] args) {

String str = null;

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

}

}

 What exception is thrown? Why does it occur?

**O/P**

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

at Main.main(Main.java:4)

// the variable **str** is initialized to **null**. Exception occurs in Java when a program attempts to use an object reference that has not been initialized.

**-------------------------------------------------------------------------------------------------------------------------------------**

Snippet 14:

public class Main {

public static void main(String[] args) {

double num = "Hello";

System.out.println(num);

}

}

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

**O/P**

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

double num = "Hello"

// Java enforces type constraints to ensure that operations are performed with compatible types, and it requires explicit type conversion where needed.

**-------------------------------------------------------------------------------------------------------------------------------------**

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);

}

}

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

**O/P**

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

int result = num1 + num2;

**//** we can handle different datatypes bys using correct datatype and typecasting.

**-------------------------------------------------------------------------------------------------------------------------------------**

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?

**O/P**

**2.0**

**//**output is what I expected.

**-------------------------------------------------------------------------------------------------------------------------------------**

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?

**O/P**

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

int result = a \*\* b;

//because it is invalid way of datatype input.\*\* is not incuded in java specific datatype.

**-------------------------------------------------------------------------------------------------------------------------------------**

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**

//As per BODMAS rule of mathematics.(\*and/ before +and-)

**-------------------------------------------------------------------------------------------------------------------------------------**

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?

**O/P**

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

at Main.main(Main.java:5)

// Values like INFINITY and NaN are available for floating-point numbers but not for integers. As a result, dividing an integer by zero will result in an exception.

**-------------------------------------------------------------------------------------------------------------------------------------**

Snippet 20:

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?

**O/P**

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

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

//the program does not get compiled. semicolon is used to terminate the statement.

**-------------------------------------------------------------------------------------------------------------------------------------**

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?

**O/P**

error: reached end of file while parsing

**-------------------------------------------------------------------------------------------------------------------------------------**

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?

**O/P**

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

static void displayMessage() {

^

Main.java:7: error: class, interface, enum, or record expected

}

^

2 errors

//method can not be declared inside another method.

**-------------------------------------------------------------------------------------------------------------------------------------**

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 switch statement continues execution from the matching case until it encounters a break statement.we can prevent the program from executing the default case by using break statement.

**-------------------------------------------------------------------------------------------------------------------------------------**

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?

// the switch statement continues execution from the matching case until it encounters a break statement. To ensure only the matching case is executed, and the default case is not inadvertently executed, add break statements at the end of each case block.

**-------------------------------------------------------------------------------------------------------------------------------------**

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?

**O/P**

Switch.java:4: error: selector type double is not allowed

switch(score) {

// Use an int for the Switch Expression and Use if-else Statements for 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?

**O/P**

Switch.java:8: error: duplicate case label

case 5:

// At runtime, if two cases in a switch statement have the same label, the second case will never be executed.

**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"

**I/P**

public class GradeClassifier

{

public static void main(String[] args)

{

int score = 90;

if (score >= 90)

{

System.out.println("A");

}

else if (score >= 80)

{

System.out.println("B");

}

else if (score >= 70)

{

System.out.println("C");

}

else if (score >= 60)

{

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.

import java.util.Scanner;

public class DayOfWeek {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a number (1-7) for the day of the week: ");

int day = scanner.nextInt();

switch (day) {

case 1:

System.out.println("Monday");

switchToWeekday();

break;

case 2:

System.out.println("Tuesday");

switchToWeekday();

break;

case 3:

System.out.println("Wednesday");

switchToWeekday();

break;

case 4:

System.out.println("Thursday");

switchToWeekday();

break;

case 5:

System.out.println("Friday");

switchToWeekday();

break;

case 6:

System.out.println("Saturday");

switchToWeekend();

break;

case 7:

System.out.println("Sunday");

switchToWeekend();

break;

default:

System.out.println("Invalid input! Please enter a number between 1 and 7.");

}

scanner.close();

}

private static void switchToWeekday() {

System.out.println("It's a weekday.");

}

private static void switchToWeekend() {

System.out.println("It's a weekend.");

}

}

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.

**I/P**

import java.util.Scanner;

public class DiscountCalculator {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the total purchase amount: Rs. ");

double totalAmount = scanner.nextDouble();

System.out.print("Do you have a membership card (yes/no)? ");

String hasMembershipCard = scanner.next().trim().toLowerCase();

double discountPercentage;

if (totalAmount >= 1000) {

discountPercentage = 20;

} else if (totalAmount >= 500) {

discountPercentage = 10;

} else {

discountPercentage = 5;

}

if (hasMembershipCard .equals("yes")) {

discountPercentage += 5;

}

double discountAmount = (discountPercentage / 100) \* totalAmount;

double finalAmount = totalAmount - discountAmount;

System.out.println("Discount Percentage: " + discountPercentage + "%");

System.out.println("Discount Amount: Rs. " + discountAmount);

System.out.println("Final Amount to Pay: Rs. " + finalAmount);

scanner.close();

    }

}