**Answer of assignment day 2 (logic building)**

-----------------------------

**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 find-->

**1.**error: class Main is public, should be declared in a file named Main.java

public class Main {

**Snippet 2:**

**public class Main {**

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

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

**}**

**}**

 What happens when you compile and run this code?

**2.** 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?**

**3.** 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---> The keyword void in java tells that the main method won't return any value

**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?**

**4.** Main method not found in class Main, please define the main method as:

public static void main(String[] args)

Why is String[] args needed?-->

In Java, String[] args is used in the main method to accept command-line arguments.

When you run a Java program, you can pass arguments to it from the command line.

These arguments are received as an array of String objects in the main method,

allowing the program to use them for various purposes, such as configuring behavior or input parameters.

**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?**

**5.** output🡪Main method with String[] args

Can you have multiple main methods?

--> Yes, you can have multiple main methods in a Java class, but only the main(String[] args) method is recognized and executed by the Java Virtual Machine (JVM) as the entry point of the application

 The JVM only calls main(String[] args) to start the application.

 The main(int[] args) method is an example of method overloading and can be called explicitly from other code.

What do you observe?

--> there are two main are there one is main class 2nd one is overloaded with different parameters class which is acceptable . but it take only main(String[] args) that one method

**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?**

**6.**

Day2\_Snippet4.java:3: error: cannot find symbol

int x = y + 10;

^

symbol: variable y

location: class Main

What error occurs?

cannot find symbol y

Why must variables be declared?

The main purpose of variable declaration is to store the required data in the memory location in the form of variables

**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?**

**7.** error: incompatible types: String cannot be converted to int

int x = "Hello";

Why does Java enforce type safety?

Java enforces type safety to prevent errors and ensure that operations are performed on compatible data types.

This helps catch mistakes at compile-time rather than runtime, reducing bugs and improving code reliability.

Type safety ensures that variables are used in ways consistent with their declared types, which makes the code more predictable and easier to maintain.

**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?**

**8.**

What syntax errors are present?

-->parenthesis missing after double quotes"

How do they affect compilation?

error: ')' or ',' expected

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

**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?**

**9.**

What error occurs?

error: not a statement

int class = 10;

^

Day2\_Snippet4.java:3: error: ';' expected

int class = 10;

^

Day2\_Snippet4.java:3: error: <identifier> expected

int class = 10;

^

Day2\_Snippet4.java:4: error: illegal start of expression

System.out.println(class);

^

Day2\_Snippet4.java:4: error: <identifier> expected

System.out.println(class);

^

5 errors

Why can't reserved keywords be used as identifiers?

Reserved keywords in Java can't be used as identifiers because they are special words that the language uses to perform specific tasks

If you could use these keywords as names for your own variables or methods, it would confuse the compiler

**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?**

**10.**

Day2\_Snippet4.java:9: error: non-static method display() cannot be referenced from a static context

display();

^

Day2\_Snippet4.java:10: error: non-static method display(int) cannot be referenced from a static context

display(5);

^

3 errors

Is method overloading allowed?

--->yes

**Snippet 11:**

**public class Main {**

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

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

**System.out.println(arr[5]);**

**}**

**}**

**11.**What runtime exception do you encounter?

Exception in thread "main" java.lang.Error: Unresolved compilation problem

Why does it occur?

due to class name as Main

**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?**

**12**. output – infinite loop

How can you avoid infinite loops?

by giving terminating condition

**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?**

**13.**

What exception is thrown?

Exception in thread "main" java.lang.NullPointerException

Why does it occur?

Cannot invoke "String.length()" because "str" is null at Main.main(Main.java:4)

**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?**

**14.**

What compilation error occurs?

error: incompatible types: String cannot be converted to double

double num = "Hello";

Why does Java enforce data type constraints?

Java enforces data type constraints to ensure that operations are performed on compatible data types, preventing errors and ensuring type safety

**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?**

**15.** error: incompatible types: possible lossy conversion from double to int

int result = num1 + num2;

How should you handle different data types in operations?

by using 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?**

**16.**

What is the result of this operation?

2.0

**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?**

**17.**compilation 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 Java does not have an exponentiation operator built into its syntax

**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?**

**18.**

output --> 20

How does operator precedence affect the result?

Operators with higher precedence are evaluated before operators 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?**

**19.**

What runtime exception is thrown?

-->Exception in thread "main" java.lang.ArithmeticException: / by zero at Main.main(Main.java:5)

Why does division by zero cause an issue in Java?

-->This is because integer division by zero is mathematically undefined and cannot be represented as a finite integer value.

**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?**

**20.**

What syntax error occurs?

--> expected semicolon ;

How does the missing semicolon affect compilation?

-->missing semicolon causes a compilation error because semicolons are used to terminate statements

Without them, the compiler cannot determine where one statement ends and another begins

**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?**

**21.**

What does the compiler say about mismatched braces?

--> syntax 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?**

**22.**

syntax error: illegal start of expression

static void displayMessage() {

^

Day2\_Snippet4.java:7: error: class, interface, enum, or record expected

Can a method be declared inside another method?

In Java, you cannot declare a method inside another method. Java requires that all methods be declared at the class level, not within other 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?**

**23.**

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

the program from executing the default case?

-->default case print after value is 2 because in code there is no break statement after value is 2

-->\*Ensure that all possible values are covered by case labels.

\*Use conditional logic within the default case if it is necessary but should only execute under certain conditions.

\*Use Input Validation

**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?**

**24.**

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?

-->If there are no break statements to terminate the case, the execution "falls through" to the next case

-->In this situation, the break statement prevents the fall-through by exiting the switch block after executing the matched case code

**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?**

**25.**

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?

-->The code does not compile because the switch statement in Java does not support double data types.

-->The switch statement only accepts integral types byte, short, int, char and enumerated types (enum) as well as String

-->modified the code to make it work as replace double with int

**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?**

**26.**

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 compiler complains because you have duplicate case labels in the switch statement

Each case label must be unique

-->When you have two identical case labels in the same switch block, the Java compiler will generate a compilation error