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?**

Main method not found in class main.Static Keyword is missing

The main method must be static because it is called by jvm.

**Corrected code:**

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?**

**Corrected code:**

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

public class Demo1 {

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?**

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

public static void main(String[] args).

Main method in java must return void not int . aslo main method should not have any return value.

Main method is entry point of java ,the main method in java uses void because it serves as a starting point of program.

**Corrected code:**

public class Three {

public static void main(String[] args) {

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

}

}

**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?**

The code compiled successfully, but it not run as expected.it looks for main method.

**String args[] is Standard Entry Point** in java. The JVM requires the main method to have the exact public static void main(String[] args) to identify it as the entry point of the program.

**Corrected code:**

public class Fourth {

public static void main() {

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

}

}

**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 define multiple main methods in a Java class. However, only one specific main method is used by the JVM to start the application.

This method is the entry point of the Java application and must be present for the JVM to execute the program.

**Corrected code:**

public class Fifth {

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");

}

}

**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?**

the code will result in a compilation error.The error occurs because the variable y is used before it has been not declared and initialized.

Variable must be use because it Specifies the data type for type-checking.

**Corrected code:**

public class Six {

public static void main(String[] args) {

int y = 5;

int x = y + 10;

System.out.println(x);

}

}

**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 is incompatible types and String cannot be converted to int

int x = "Hello";

Java enforces type safety to prevent errors and ensure reliability by making sure that variables are used consistently according to their declared data types.

**Corrected code:**

public class Seven {

public static void main(String[] args) {

String x = "Hello";

System.out.println(x);

}

}

**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?**

**Missing Parenthesis** is there. The original

System.out.println("Hello, World!" is missing the closing ).

Missing or incorrect parentheses cause syntax errors, preventing the code from compiling. The compiler generates an error message indicating the issue, and the program won't run until the error is fixed.

**Corrected code:**

public class Eight {

public static void main(String[] args) {

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?**

error: ';' expected int class = 10;

The code is not compile because it uses class as a variable name, which is not allowed in Java. The word class is a reserved keyword in Java, used to define classes, and cannot be used as an identifier for variables, methods, or other elements.Reserved keywords in Java can't be used as identifiers because they are essential to the language's structure and syntax.

**Corrected code:**

public class Nine{

public static void main(String[] args) {

int number = 10;

System.out.println(number);

}

}

**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?**

Error is showing that non-static method display() cannot be referenced from a static context.

Yes, **method overloading** is allowed in Java. Method overloading occurs when you define multiple methods in the same class with the same name but different parameter lists. The difference can be in the number of parameters, the type of parameters, or the order of parameters.

**Corrected code:**

public class Ten {

public static void display() {

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

}

public static void display(int num) {

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

}

public static void main(String[] args) {

Ten obj = new Ten();

display();

display(5);

}

}

**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:** ArrayIndexOutOfBoundsException.

The code tries to access index 5 of an array with only 3 elements (valid indices are 0, 1, and 2).

Because of The index 5 is out of the array’s bounds, which causes this exception at runtime.

**Corrected code:**

public class Eleven {

public static void main(String[] args) {

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

System.out.p rintln(arr[2]);

}

}

**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.**

After running the code it into infinite loops .

To correct the infinite loop, you should include a condition that will eventually be met to exit the loop.

Ex:int count = 0;

while (count < 5)

**Corrected code:**

public class Twelve {

public static void main(String[] args) {

int count = 0;

while (count < 5)

{

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

count++;

}

}

}

**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?**

**Exception**: NullPointerException.

**Because**The variable str is assigned null, meaning it doesn't reference any actual String object.

**Corrected code:**

public class Thirteen{

public static void main(String[] args) {

String str = "hello world";

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

}

}

**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?**

In this code, you are trying to assign a String value Hello to a double variable num. Java enforces data type constraints to ensure that variables hold values that make sense for their declared types.

**Corrected code:**

public class Main {

public static void main(String[] args) {

String num = "Hello";

System.out.println(num);

}

}

**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?**

The variable num1 is an int, and num2 is a double.

**Corrected code:**

public class Fifteen{

public static void main(String[] args) {

int num1 = 10;

double num2 = 5.5;

double result= num1 + num2;

System.out.println(result);

}

}

**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?**

num / 4 where num is an int and 4 is also an int.

When dividing two int values, Java performs integer division, which discards the fractional part and returns an int result.

**Corrected code:**

public class Sixteen{

public static void main(String[] args) {

int num = 10;

double result = num / 4;

System.out.println(result);

}

}

**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?**

illegal start of expression int result = a \*\* b; Error is operator \*\* cannot be applied to int, int.

Java does not support the \*\* operator for exponentiation. The \*\* operator is not a valid operator in Java, which is why the compiler throws an error.

Java supports basic arithmetic operators such as +, -, \*, and / for addition, subtraction, multiplication, and division, respectively. However, it does not include an operator for exponentiation.

**Corrected code:**

public class Seventeen {

public static void main(String[] args) {

int a = 10;

int b = 5;

int result = a \*\* b;

System.out.println(result);

}

}

**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?**

Output is 20. In Java, operator precedence determines the order in which operations are performed. The precedence rules are similar to standard arithmetic:

Multiplication \* has higher precedence than Addition +.

**Corrected code:**

public class Eighteen {

public static void main(String[] args) {

int a = 10;

int b = 5;

int result = a + b \* 2;

System.out.println(result);

}

}

**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: ArithmeticException

Message: division by zero

Division by zero is mathematically undefined. In Java, as in most programming languages, dividing an integer by zero does not produce a valid result.

**Corrected code:**

public class Ninteen {

public static void main(String[] args) {

int a = 10;

int b = 0;

if (b != 0){

int result = a / b;

System.out.println(result);

}else {

System.out.println("Error: Division by zero is not allowed.");

}

}

}

**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?**

error: ';' expected System.out.println("Hello, World")

A missing semicolon causes a syntax error in Java, preventing compilation. The compiler expects semicolons to terminate statements, and without them, it cannot parse the code correctly, leading to error messages like `';' expected`.

**Corrected code:**

public class Twenty {

public static void main(String[] args) {

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

}

}

**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?**

**Error Message**: The compiler will indicate a "reached end of file while parsing" or "unexpected end of file" error.

**Corrected code:**

public class Twentyone {

public static void main(String[] args) {

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?**

Error is class, interface, enum, or record expected

public classTwentytwo.

In Java, a method cannot be declared inside another method.

**Static Method Inside** main **Method,** In Java, cannot define a method like displayMessage inside another method like main. Methods should be defined at the class level, not within other methods.

**Corrected code:**

public class Twentytwo {

static void displayMessage() {

System.out.println("Message");

}

public static void main(String[] args) {

displayMessage();

}

}

**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 evaluates the expression value, which is 2 in this case.When it finds a matching case, it executes the code associated with that case. after matching a case, the switch statement will continue to execute all subsequent cases and the default case unless there is a break statement to exit the switch.

**Corrected code:**

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;

}

}

}

**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?**

1. "Level 1", "Level 2", "Level 3", and "Unknown level" are printing because

of not using Break keyword after each statement.

2.The break statement is used to exit the switch block once the matching case is executed.

**Corrected code:**

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;

}

}

}

**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?**

1.The issue lies in the fact that the switch statement in Java does not support double (or float) as a

valid type for the expression being switched on.

2.Use an int Type Instead of double to modify it

**Corrected code:**

public class Switch {

public static void main(String[] args) {

int score = 85;

switch(score) {

case 100:

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

break;

case 85:

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

break;

default:

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

}

}

}

**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?**

**Corrected code:**

public class Twentysixth {

public static void main(String[] args) {

int number = 5;

switch(number) {

case 5:

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

break;

case 10:

System.out.println("This is another case 10");

break;

default:

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

}

}

}

**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"

**Code:**

class GradeClassification

{

public static void main (String args[]){

int score=75;

if(score>=90){

System.out.println("Grade A");

}

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

System.out.println("Grade B");

}

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

System.out.println("Grade C");

}

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

System.out.println("Grade D");

}

else{

System.out.println("Grade 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.**

**Code:**

public class DaysOfTheWeek {

public static void main(String[] args) {

int day = 3;

switch(day) {

case 1:

System.out.println("Monday ");

break;

case 2:

System.out.println("Tuesday ");

break;

case 3:

System.out.println("Wednesday ");

break;

case 4:

System.out.println("Thursday ");

break;

case 5:

System.out.println("Friday ");

break;

case 6:

System.out.println("Saturday");

break;

case 7:

System.out.println("Sunday ");

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 if else to check if division by zero is attempted and display an error message.**

**Code:**

import java.util.Scanner;

public class BasicCalculator {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the first number: ");

double num1 = scanner.nextDouble();

System.out.print("Enter the second number: ");

double num2 = scanner.nextDouble();

System.out.print("Enter an operator (+, -, \*, /): ");

char operator = scanner.next().charAt(0);

switch (operator) {

case '+':

System.out.println("Result: " + (num1 + num2));

break;

case '-':

System.out.println("Result: " + (num1 - num2));

break;

case '\*':

System.out.println("Result: " + (num1 \* num2));

break;

case '/':

if (num2 != 0) {

System.out.println("Result: " + (num1 / num2));

} else {

System.out.println("Error: Division by zero.");

}

break;

default:

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

break;

}

scanner.close();

}

}

**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%.**

**Code:**

class Discount {

public static void main(String args[]){

Scanner sc = new Scanner(System.in);

System.out.println("Enter price ");

int price =sc.nextInt();

System.out.println("Enter 1 if membership present else 0");

int membership = sc.nextInt();

if(price >=1000){

if(membership==1){

System.out.println("Discount is : " + ((price\*25)/100));

}else{

System.out.println("Discount is : "+(price\*20)/100);

}

}else if(price >=500 && price<=999){

if(membership==1){

System.out.println("Discount is : "+(price\*15)/100);

}else{

System.out.println("Discount is : "+(price\*10)/100);

}

}else{

if(membership==1){

System.out.println("Discount is : "+(price\*10)/100);

}else{

System.out.println("Discount is : "+(price\*5)/100);

}

}

}

}

**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.**

**Code:**

class FailPass{

public static void main(String[] args){

int c=10,cpp=20,java=50,count=0;

if(c>40){

count+=1;

}

if(cpp>40){

count+=1;

}

if(java>40){

count+=1;

}

if(count==3){

System.out.println("passed in all 3 subjects");

}else if(count==2){

System.out.println("failed in all 1 subjects");

}else if(count==1){

System.out.println("failed in all 2 subjects");

}

else{

System.out.println("failed in all subjects");

}

}

}