Section 1.

Snippet 1:

public class Main {

public void main(String[] args) {

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

}

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

public static void main(String[] args)” . static keyword is needed when defining main ().

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!");

}

}

Solution -- we get an error when running the above code .Error-“ Main method not found in class Main, please define the main method as: public static void main(String[] args) “. public access specifier is needed when defining main ().

Corrected code --

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;}

Error- return type of main() should be void instead of int. It should not return anything because when main() terminates the program terminates too .So doesn’t make sense main() to return anything.JVM doesn’t expect anything from main().

Corrected code --

public class Main {

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!");

}

}

Error- “command line arugments are missing from main() definition “.

Corrected Code—

public class Main {

public static void main(String args[]) {

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"); }

}

Solution-- yes, there can be multiple main() in a java program .Only the original main() is called , overloaded main() is not called. To invoke it we need to call it from the original main().

Snippet 6:

public class Main {

public static void main(String[] args) {

int x = y + 10;

System.out.println(x);

}

}

Error-- variable y is not declared and initialized..

Variables must be declared as it tells the compiler about the type of data it will hold and how much memory it needs to be allocated.And to use the variable it must also be initialized to some value.

Corrected Code—

public class Main {

public static void main(String[] args) {

int y=0;

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);

} }

Error- ‘x’ is int data type variable and it has been assigned a string value .

Corrected Code—

public class Main {

public static void main(String[] args) {

int x = 2;

System.out.println(x);

} }

Snippet 8:

public class Main {

public static void main(String[] args) {

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

}

}

Error- parenthesis and semicolon after ("Hello, World!" are missing.

Corrected Code—

public class Main {

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);

}

Error- class is a reserved keyword cannot be used as a variable name because its meaning is already defined.

Corrected code—

public class Main {

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);

}

}

Solution - The code will throw error when compiled because non-static methods- display() and display(int num) are being accessed by static method main().

Yes, method overloading is allowed in java .Using method overloading we can use same name for different methods provided we use different parameters for them .

Snippet 11:

public class Main {

public static void main(String[] args) {

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

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

}

}

Solution- the above code throws a runtime error-“index out of bounds” because the element number being accessed exceeds the number of elements of the array.

Corrected Code—

public class Main {

public static void main(String[] args) {

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

System.out.println(arr[2]);

}

}

Snippet 12:

public class Main {

public static void main(String[] args) {

while (true) {

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

}

}

}

Solution- the code runs infinitely as the above the condition in while loop will always remain true.

We can avoid infinite loop by using break statement in the loop.

Corrected Code—

public class Main {

public static void main(String[] args) {

while (true) {

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

break;

}

}

}

Snippet 13:

public class Main {

public static void main(String[] args) {

String str = null;

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

}

}

Solution- the above code throws an error-“cannot invoke string.length() as the string is null”.It occurs because the function string.length() is used to find out length of the string but the string “str” has no value associated with it.

Corrected Code—

public class Main {

public static void main(String[] args) {

String str = “hello”;

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

}

}

Snippet 14:

public class Main {

public static void main(String[] args) {

double num = "Hello";

System.out.println(num);

}

}

Solution-- the above code throws “incompatible type” error because a string value is being stored in double data type. Java enforces data type constraint to ensure the reliability and accuracy of data and get expected results.

Corrected Code—

public class Main {

public static void main(String[] args) {

double num = 56.234;

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);

}

}

Solution-- the error for above code is “incompatible types: possible lossy conversion from double to int” .

This is because the num2 is of double data type and after its addition with num1(an int data type) , the result will be of “double” but it is being stored in another variable called result which is “int”.

In operations a value of higher size cannot be stored in a data type with lower size. Always choose the higher size data type to store value in operations.

Corrected Code—

public class Main {

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);

}

}

Solution-- The result of above operation is 2 which is an integer. Ideal result is 2.5 but the code doesn’t produce it because both the value involved in operation are “int” .To get “double” value, both or either of the number should be written in double format or should be type casted.

Corrrected Code—

public class Main {

public static void main(String[] args) {

int num = 10;

double result = num / 4.0;

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);

}

}

Error—“ error: illegal start of expression” On line 5 .Because \*\* is not valid in Java instead Math.pow(base, power) is used under Math class for exponent operations.

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?

Solution-- output of the code is 20. Operator precedence affects the order in which operations are performed. In the above code if the expression was a\*b+2 result would have been =52.

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?

* Solution 19- an arithmetic exception is thrown –“/ by zero ”. The result depends on type of expression -In case of double/float division, the output is Infinity, the basic reason behind that it implements the floating point arithmetic algorithm which specifies a special values like “Not a number” OR “infinity” for “divided by zero cases” as per IEEE 754 standards.In case of integer division, it throws ArithmeticException.

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?

Solution 20-- above code produces syntax error – “';' expected” at line 3.

This is because syntactically every statement in java program should end with ‘;’. At compile time all the syntax and semantics are checked and a missing semicolon generates a compile time error.

Corrected Code—

public class Main {

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?

Solution 21 —the above code generates error: “reached end of file while parsing “ .It is a compile time syntax error.

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?

Solution 22—Syntax error-“ illegal start of expression” .No , in java a method cannot 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?

Solution 23—default case is printed after “Value is 2” because case 2 doesn’t include a “break” statement. Hence the program runs and executes all the cases after case2 . We can prevent

the program from executing the default case by including break statement in every 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?

Solution 24—In java the switch statement executes all the statements after the first case match until a break statement is reached.Hence , after level 1 it prints "Level 1", "Level 2", "Level 3", and "Unknown level".

break statement should be included in all cases so that only the case that matches first gets 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?

Solution 25—the code doesn’t compile ,error- “incompatible types: possible lossy conversion from double to int” because the expression in switch statement is a double value and that is not allowed in java.

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?

Solution 26—in java, the switch statement does not allow duplicate case values. Each case label must be unique within the same switch block. This ensures that the program can accurately determine which code block to execute based on the switch expression’s value.