1)public class Main {

public void main(String[] args) {

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

}

}

// In the above code main method is not static. Static means that you can run this method without creating an instance of main.

2) public class Main {

static void main(String[] args) {

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

}

}

// In the above code main method is missing we should write it as public static void main(String args[]). Public again means that anyone can access it.

3) public class Main {

public static int main(String[] args) {

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

return 0;

}

}

// In the above code the return type of value should always be a void.

4) public class Main {

public static void main() {

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

}

}

// the above code throws an error because input to the main method is not given.

// where String args[] in java is an array of type java.lang.String class that stores java command line arguments.

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");

}

}

// yes we can have multiple main method. But it only gives output of first main method after execution.

6) public class Main { public static void main(String[] args) {

int x = y + 10; System.out.println(x);

}

}

// It gives an error because we have not initialised the y here.

7) public class Main { public static void main(String[] args) {

int x = "Hello";

System.out.println(x);

}

}

// It gives an error because string cannot be converted to int we should take care of typecasting.

8) public class Main { public static void main(String[] args) {

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

}

}

// It gives an syntax error because it is a case sensitive like c language.

9) public class Main { public static void main(String[] args) {

int class = 10;

System.out.println(class);

}

}

// it gives error because reserved identifiers cannot be used as variable name it do not have any user defined meaning. Decreased the readability of the program.

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);

}

}

// it gives an error that non-static method display() or display(5) cannot be referenced from a static context. Overloading is not allowed in java.

11) public class Main {

public static void main(String[] args) {

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

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

}

}

// It gives Index 5 out of bounds for length 3. Because we have declared only 3 index array length and want output of 5 index array length.

12) public class Main {

public static void main(String[] args) {

while (true) {

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

}

}

}

// It prints infinite loop continuously

/\* Use clear relational operators to prevent unintentional infinite loops. Use of break statements can make code harder to understand, but in some cases, they can prevent infinite loops.

13) public class Main {

public static void main(String[] args) {

String str = null;

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

}

}

// It gives null pointer exception because we have declared string variable as a null which is not correct as per java protocols.

14) public class Main {

public static void main(String[] args) {

double num = "Hello";

System.out.println(num);

}

}

// It gives incompatible type error of string cannot converted to double.to improve code readability and catch type-related errors at compile time.

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);

}

}

// It gives incompatible type error of double to int is not converted. Whereas we can stored result in double but not in int.

16) public class Main {

public static void main(String[] args) {

int num = 10;

double result = num / 4;

System.out.println(result);

}

}

// It gives output as 2.0 in double.

17) public class Main {

public static void main(String[] args) {

int a = 10;

int b = 5;

int result = a \*\* b;

System.out.println(result);

}

}

// It gives expression error.

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); } }

// 20 is the ouput. Precedence will be from left to right.

19) public class Main {

public static void main(String[] args) {

int a = 10;

int b = 0;

int result = a / b;

System.out.println(result); } }

// it gives arithmetic exception error because infinity which is inappropriate as result.

20) public class Main {

public static void main(String[] args) {

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

// If semicolon is not given in Java program, syntax will occur during the compile time. The set of grammatical rules of a programming language for writing statements of the computer program is known as syntax of the language. Thus it occurs syntax of program is not syntactically correct.

21) public class Main {

public static void main(String[] args) {

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

// Missing closing brace here }

// If the code has mismatched braces then it will always fails to compile the program.

22) public class Main {

public static void main(String[] args) {

static void displayMessage() {

System.out.println("Message"); } } }

// Return type is void so it gives error.

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"); } } }

// break statements should be used after print of every case.

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"); } } }

// use break after every case.

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!"); } } }

// incompatible error of double to int conversion not possible. Type casting should be done but double is not used in switch case.

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");

}

}

}

// It gives error because switch case has rules which we need to follow that is we cannot create duplicate cases.