

Section 1: Error-Driven Learning in Java

Snippet 1:

```
public class Main {  
    public void main (String [] args) {  
        System.out.println("Hello, World!");  
    }  
}
```

Error:

```
E:\CDAC\java\Day 2>java Main  
Error: Main method is not static in class Main, please define the main method as:  
    public static void main(String[] args)
```

The static keyword is missing in the code.

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!");  
    }  
}
```

Error:

```
E:\CDAC\java\Day 2>java Main  
Error: Main method not found in class Main, please define the main method as:  
    public static void main(String[] args)  
or a JavaFX application class must extend javafx.application.Application
```

The public keyword is missing in the code. If main is not public, the Java Virtual Machine (JVM) won't be able to access it, and the program will not run.

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:

```
ER (E01) (java) (say) E java main  
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)
```

As the main method is the entry point of a Java application. The Java virtual machine (JVM) does not require a return value from main, so it must be declared with void.

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:

```
Error: Main method not found in class Main, please define the main method as:  
    public static void main(String[] args)  
or a JavaFX application class must extend javafx.application.Application
```

String[] args allows users to pass arguments from the command line when running the program.

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");  
    }  
}
```

Java allows multiple methods named main in the same class, as long as they have different parameter types.

So, there is no error in this code. But it will print standard main method having String args[]

```
E:\CDAC\java\Day 2>java Main  
Main method with String[] args
```

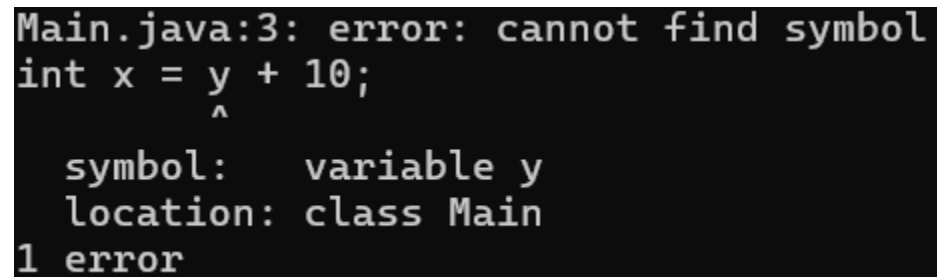
Corrected code:

```
public class Main {  
    public static void main (String[] args) {  
        System.out.println("Main method with String[] args");  
    }  
}
```

Snippet 6:

```
public class Main {  
    public static void main(String[] args) {  
        int x = y + 10;  
        System.out.println(x);  
    }  
}
```

Error:

A screenshot of a Java compiler error message. The text is white on a black background. It shows the file name and line number: 'Main.java:3: error: cannot find symbol'. Below this, the code snippet 'int x = y + 10;' is shown with a caret '^' pointing to the variable 'y'. The error details are listed: 'symbol: variable y' and 'location: class Main'. At the bottom, it says '1 error'.

The variable y is not declared before being used.

Corrected code:

```
public class Main {  
    public static void main(String[] args) {  
        int y = 6;  
        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:

```
E:\CDAC\java\Day 2>javac Main.java  
Main.java:3: error: incompatible types: String cannot be converted to int  
    int x = "Hello";  
           ^  
1 error
```

Error is occurred because we declared x as an integer(int) , but assigned a String value to it.

Corrected code:

```
public class Main {  
    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!"  
    }  
}
```

Error:

```
E:\CDAC\java\Day 2>javac Main.java  
Main.java:3: error: ')' expected  
    System.out.println("Hello, World!"  
                        ^  
1 error
```

There are two syntax errors: the first is a missing closing parenthesis, and the second is a missing semicolon at the end.

Corrected output:

```
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

```
E:\CDAC\java\Day 2>javac Main.java
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: <identifier> expected
System.out.println(class);
                  ^
Main.java:4: error: illegal start of type
System.out.println(class);
                  ^
Main.java:4: error: <identifier> expected
System.out.println(class);
                  ^
Main.java:6: error: reached end of file while parsing
}
^
7 errors
```

In this code class is a reserved keyword in Java and cannot be used as a variable name.

Corrected code:

```
public class Main {
    public static void main(String[] args) {
        int n = 10;
        System.out.println(n);
    }

}
```

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);  
    }  
}
```

```
E:\CDAC\java\Day 2>javac Main.java  
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
```

The display() methods are instance methods (non-static) because they do not have the static keyword.

Corrected code:

```
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) {  
        Main obj = new Main();  
        obj.display();  
        obj.display(5);  
    }  
}
```


Snippet 11:

```
public class Main {  
    public static void main(String[] args) {  
        int[] arr = {1, 2, 3};  
        System.out.println(arr[5]);  
    }  
}
```

Error:

```
E:\CDAC\java\Day 2>java Main  
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException:  
    at Main.main(Main.java:4)
```

The error `ArrayIndexOutOfBoundsException` occurs because the code tries to access index 5, which does not exist in the array. In Java, the index starts from 0, and in this code, there are only up to 2 indexes.

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");  
  
        }  
}
```

```
}
```

```
}
```

Error:

It will print an infinite loop infinitely.

Corrected code:

```
public class Main {
```

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

```
        int a = 5;
```

```
        while (a<=10) {
```

```
            System.out.println(a);
```

```
            a++;
```

```
        }
```

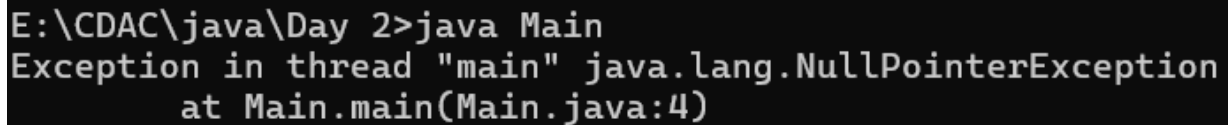
```
    }
```

```
}
```

Snippet 13:

```
public class Main {  
  
    public static void main(String[] args) {  
  
        String str = null;  
  
        System.out.println(str.length());  
  
    }  
  
}
```

Error:

A screenshot of a terminal window with a black background and white text. The text shows a command being executed and the resulting error message.

```
E:\CDAC\java\Day 2>java Main  
Exception in thread "main" java.lang.NullPointerException  
    at Main.main(Main.java:4)
```

As str is null, it does not reference a valid String object, which results in a NullPointerException when calling length().

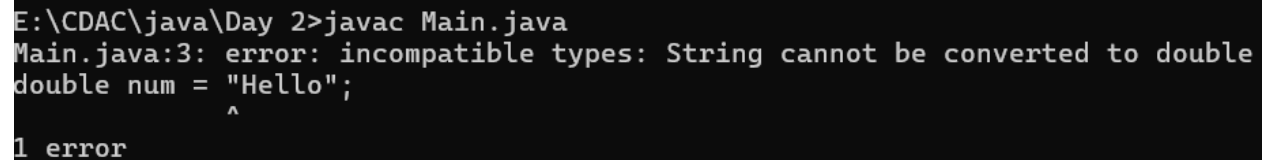
Corrected code:

```
public class Main {  
  
    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);  
  
    }  
  
}
```

Error:



```
E:\CDAC\java\Day 2>javac Main.java  
Main.java:3: error: incompatible types: String cannot be converted to double  
double num = "Hello";  
              ^  
1 error
```

The variable num is declared as a double. A double can store decimal numbers but the code tries to assign a String to num:

Corrected code:

```
public class Main {  
  
    public static void main(String[] args) {  
  
        double num = 4.0;  
  
        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);  
  
    }  
  
}
```

Error:

```
E:\CDAC\java\Day 2>javac Main.java  
Main.java:5: error: incompatible types: possible lossy conversion from double to int  
int result = num1 + num2;  
                  ^  
1 error
```

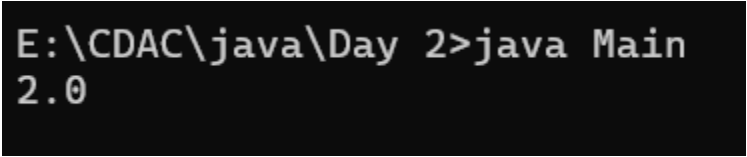
Since double cannot be directly assigned to an int , because int cannot store decimal values

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);  
  
    }  
  
}
```



```
E:\CDAC\java\Day 2>java Main  
2.0
```

The Output is 2.0 Instead of 2.5 because num is an int (10) and 4 is also treated as an int.

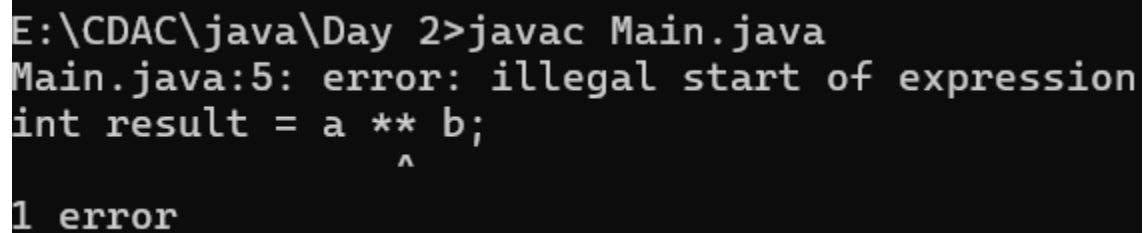
Corrected code:

```
public class Main {  
  
    public static void main(String[] args) {  
  
        int num = 10;  
  
        double result = (double) 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);  
  
    }  
  
}
```

Error:



The screenshot shows a terminal window with a black background and white text. The command 'javac Main.java' has been executed. The output shows an error at line 5: 'error: illegal start of expression' pointing to the '**' operator in the line 'int result = a ** b;'. Below the error message, it says '1 error'.

```
E:\CDAC\java\Day 2>javac Main.java  
Main.java:5: error: illegal start of expression  
int result = a ** b;  
                ^  
1 error
```

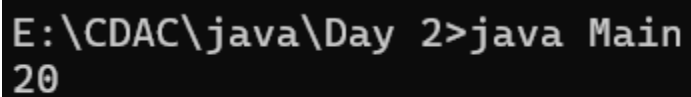
In some languages like Python, ** is used to calculate powers. However, Java does not have a built-in ** operator.

Corrected code:

```
public class Main {  
  
    public static void main(String[] args) {  
  
        int a = 10;  
  
        int b = 5;  
  
        double result = Math.pow(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);  
  
    }  
  
}
```



```
E:\CDAC\java\Day 2>java Main  
20
```

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);  
  
    }  
  
}
```


Error

```
E:\CDAC\java\Day 2>java Main
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at Main.main(Main.java:5)
```

Division by zero is mathematically not allied, and Java follows this mathematical principle.

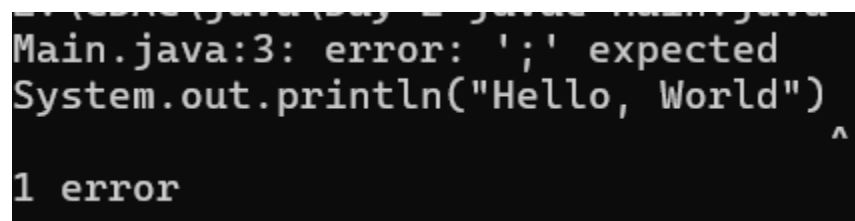
Corrected Code:

```
public class Main {
    public static void main(String[] args) {
        double a = 10.0;
        double b = 0.0;
        double result = a / b;
        System.out.println(result);
    }
}
```

Snippet 20:

```
public class Main {  
  
    public static void main(String[] args) {  
  
        System.out.println("Hello, World")  
  
    }  
  
}
```

Error:

A screenshot of a terminal window showing a Java compilation error. The text is as follows:
Main.java:3: error: ';' expected
System.out.println("Hello, World")
^
1 error
The error message indicates that a semicolon is missing at the end of the println statement on line 3.

In Java, the semicolon (;) is used to terminate statements. It tells the compiler that the current statement has ended, but when a semicolon is missing, the compiler cannot determine where the statement ends, so the syntax error occur.

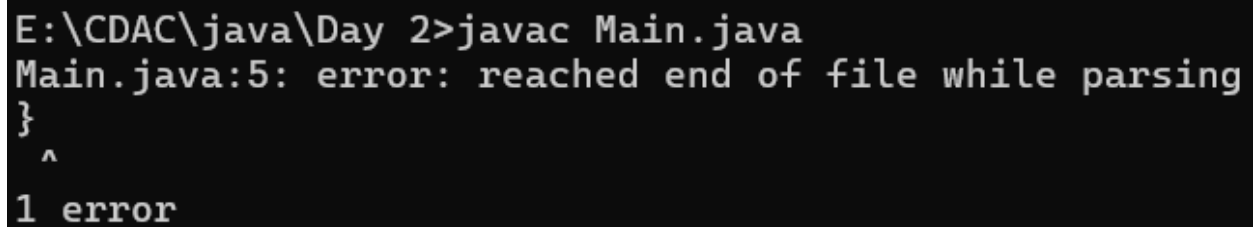
Corrected Code:

```
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  
  
    }  
}
```

Error:



A screenshot of a terminal window with a black background and white text. The text shows the command 'javac Main.java' being executed in the directory 'E:\CDAC\java\Day 2'. The output shows an error: 'Main.java:5: error: reached end of file while parsing' followed by a closing brace '}' and a caret '^' pointing to it. At the bottom, it says '1 error'.

```
E:\CDAC\java\Day 2>javac Main.java  
Main.java:5: error: reached end of file while parsing  
    }  
    ^  
1 error
```

There is a missing closing brace for the main method

Corrected Code:

```
public class Main {  
  
    public static void main(String[] args) {  
  
        System.out.println("Hello, World!");  
  
        }  
    }  
}
```

Snippet 22:

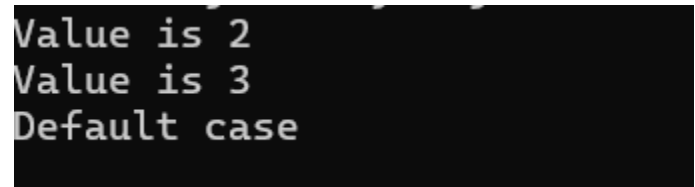
```
public class Main {  
  
    public static void main(String[] args) {  
  
        static void displayMessage() {  
  
            System.out.println("Message");  
  
        }  
  
    }  
  
}
```

Error:

```
E:\CDAC\java\Day 2>javac Main.java  
Main.java:3: error: illegal start of expression  
static void displayMessage() {  
^  
Main.java:7: error: class, interface, or enum expected  
}  
^  
2 errors
```

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");  
  
        }  
  
    }  
  
}
```



```
Value is 2  
Value is 3  
Default case
```

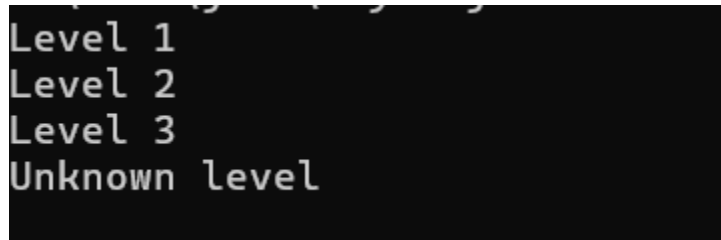
There is no break statement at the end of every case. If a break statement is not provided, it will continue to the next case even if the case does not match.

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");  
  
        }  
  
    }  
}
```



```
Level 1  
Level 2  
Level 3  
Unknown level
```

There is no break statement at the end of every case. If a break statement is not provided, it will continue to the next case even if the case does not match.

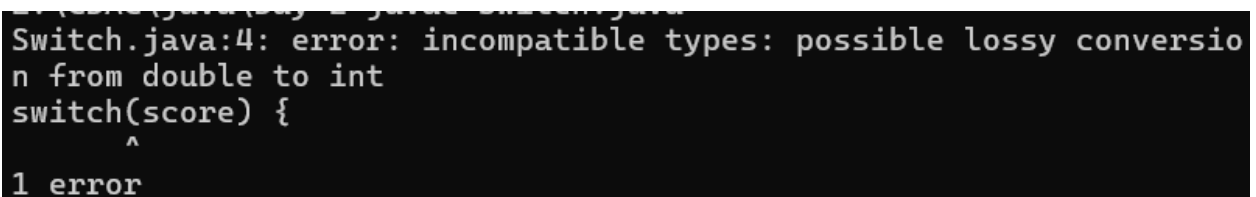
Corrected code:

```
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:

A screenshot of a Java compiler error message. The text is as follows:
Switch.java:4: error: incompatible types: possible lossy conversion
n from double to int
switch(score) {
 ^
1 error

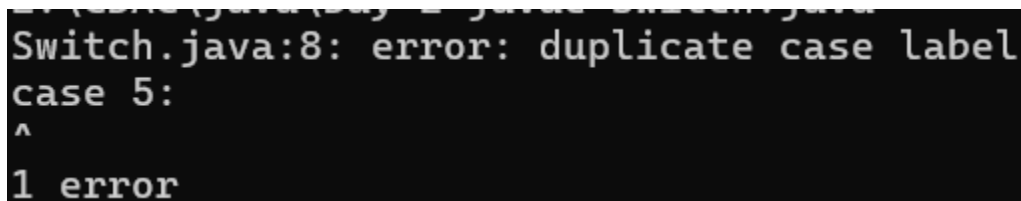
The Switch statement expects an expression for only int, char, and string datatypes. So double cannot be directly used in a switch statement .

Corrected code:

```
public class Switch {  
    public static void main(String[] args) {  
        double score = 85.0;  
        if (score == 100) {  
            System.out.println("Perfect score!");  
        } else if (score == 85) {  
            System.out.println("Great job!");  
        } else {  
            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");  
  
        }  
  
    }  
}
```

A screenshot of a terminal window showing a Java compilation error. The text is as follows:
Switch.java:8: error: duplicate case label
case 5:
^
1 error

In a switch statement, duplicate cases are not allowed because it evaluates the expression and jumps to the matching case label. If two case labels have the same value, the compiler cannot determine which one to execute.

Corrected code:

```
public class Switch {  
  
    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 5");  
  
                break;  
  
            default:  
  
                System.out.println("This is the default case");  
  
                }  
  
        }  
  
    }
```