Name: Shriniwas Vasant Pawar

Center: Kharghar

Email: pawar.shriniwas26@gmail.com

Snippet 1:

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

What error do you get when running this code?

Ans: Error: Main method is not static in class Main

Explanation: In this code snippets main method has not have static keyword. And that's why this code got error. The reason is when we mention main method as static then compiler need not to create object of that class to access it.

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?

Ans: Error: Main method not found in class Main,

Explanation: In this code snippet public is not mentioned in main method that's why compiler cant able to find main method directly.

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;
        }
}
```

What error do you encounter? Why is void used in the main method?

Ans: Error: Main method must return a value of type void in class Main,

Explanation: In above code snippet main method is declared as int it should be void in java. Because In java the default return type of main method is 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!");
}
}
```

What happens when you compile and run this code? Why is String[] args needed?

Ans: Error: Main method not found in class Main because String[] args not written in main method.

Explanation: String[] args is used to accept command line arguments. Because in java syntax we have to write String[] args in main method to accept command line arguments.

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");
```

```
        }
}
```

Can you have multiple main methods? What do you observe?

Ans: we can have multiple main methods but with different types of args command line argument. It will execute the default main method which has String[] args in it. In it method overloading is observed.

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);
  }
}
```

What error occurs? Why must variables be declared?

Ans: error: cannot find symbol y.

Explanations: we must declare variables before using it because compiler doesn't recognize variable that is call without declaration.

Corrected code:

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

```
    int y = 1;
    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?

Ans: error: incompatible types: String cannot be converted to int,

Java enforce type safety because there should not be conflict among variables datatypes which leads to error while coding in future when that variable is used.

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!"
```

```
  }
}
```

What syntax errors are present? How do they affect compilation?

Ans: error: ')' expected

If we do not write ');' at end of println or if we doesn't follow any syntax rule. code get syntax error of that type.

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);
  }
}
```

What error occurs? Why can't reserved keywords be used as identifiers?

Ans: error: Identifier expected and not a statement

We cant used reserved keywords like 'class' as identifier because if we use it then compiler cant able to recognize which is identifier and which is keyword this conflict will happen if we use it.

Corrected code:

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

```
  }
}
```

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?

Ans: when we compile this code then compiler gives error stating that you cannot reference non static method as static context. Method overloading is allowed but it should be referenced as non static method to run this code.

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 m = new Main();
    m.display();
    m.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?

Ans: It encounters ArrayIndexOutOfBounds Exception at runtime because at compile time it has no syntax error but at runtime compiler cannot access the memory space of that array index which is beyond its limit.

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");
    }
  }
}
```

What happens when you run this code? How can you avoid infinite loops?

Ans: If we run this code then infinite times "Infinite Loop" statement runs.

We can avoid infinite loops by adding a condition to loop that will be false in future and increment or decrement the variable depending on loop condition.

Corrected code:

```
public class Main {
  public static void main(String[] args) {
    int i = 1;
    while (i<=5) {
      System.out.println("Infinite Loop");
      i++;
    }
  }
}
```

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?

Ans: At compile time syntax is correct But at runtime it gives nullPointerException because String is pointed to null and memory is not allocated to this variable that's why we cannot get the length of string and this exception occurs

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);
  }
}
```

What compilation error occurs? Why does Java enforce data type constraints?

Ans: Here incompatible type error occurs it states that String cannot be converted to double. Java enforce data type constraints because there should not be conflict on datatype of variables If java doesn't enforce it then error can be occur in future while processing that variable.

Corrected code:

```
public class Main {
  public static void main(String[] args) {
    double num = 15.00;
    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?

Ans: “Incompatible types lossy conversion from double to int” this error will occur. we should type cast int to double and make datatype of result as double to avoid this error. So while writing code we should be careful about datatypes where incompatible types not occur to overcome this we should typecast variables.

Corrected code:

```
public class Main {
    public static void main(String[] args) {
        int num1 = 10;
        double num2 = 5.5;
        double result = (double) 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?

Ans: The result of this operation is 2.0. It is not the output that I expected. The correct result should be 2.5. So to achieve this result we should typecast any one variable or integer literal as double while dividing. so then we get the correct result

Corrected code:

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

Ans: Illigal start of expression occurs while compiling the code. Because there is no such operator ** in java. We can use Math.pow() method to achieve the result.

Corrected code:

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

What is the output of this code? How does operator precedence affect the result?

Ans: output of this code is 20. Here the operator which has higher precedence will execute first . In above code b * 2 will execute first then + a executed and stored in 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?

Ans: Arithmetic exception: Divide by 0 occurs here. In java we cannot divide any variable by 0. therefore this exception occurs. To avoid it we should divide it by any other integer variable.

Corrected code:

```
public class Main {
  public static void main(String[] args) {
    int a = 10;
    int b = 2;
    int result = a / b;
    System.out.println(result);
  }
}
```

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?

Ans: Syntax error “;” expected error occurs here. As per java syntax every statement should end with ‘;’ otherwise we get 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?

Ans: compiler says error: reached end of file while parsing. We should follow java syntax that every method should have opening and closing brackets. Here closing brace is missing. That's why it causes error.

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");
    }
  }
}
```

What syntax error occurs? Can a method be declared inside another method?

Ans: we get illegal start of expression error. We cannot declare a method inside another method According java rules.

Corrected code:

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

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");
    }
  }
}
```

Why does the default case print after "Value is 2"? How can you prevent the program from executing the default case?

Ans: Here first “Value is 2” is print then “Value is 3” print then “Default case” is printed. We can prevent this wrong behaviour by adding break statement to end of each case.

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");
    }
  }
}
```

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");
    }
  }
}
```

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?

Ans: it print "Level 1", "Level 2", "Level 3", and "Unknown level". This is printed because there is no break statement. The role of break statement is to go outside of switch.

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");
      }
   }
}
```

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!");
    }
  }
}
```

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?

Ans: It gives error of Incompatible types error lossy conversion from double to int.

This error occurs because swich cannot take input variables of double datatypes. To avoid this we should pass int, String or char datatype variables.

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");
      }
   }
}
```

Why does the compiler complain about duplicate case labels? What happens when you have two identical case labels in the same switch block?

Ans: The compiler complains about duplicate case labels because switch cannot handle duplicated cases because compiler gets confuse which case to execute.