OBJECT ORIENTED PROGRAMMING USING JAVA

OUTLINE

- Loops
- While Loop
- Do-While Loop
- For Loop
- Nested Loop
- Break and Continue in Loop

LOOPS

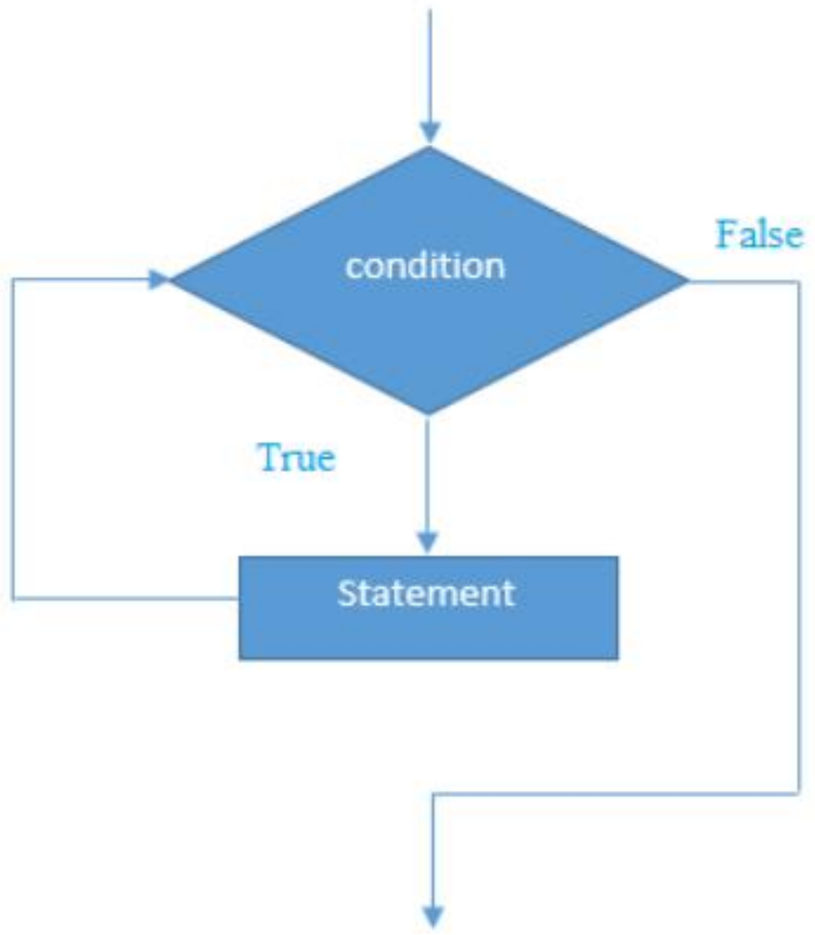
- Loops can execute a set lines or statements as long as the specified condition is satisfied.
- It mainly saves time.
- There are mainly two types of loops:
  - **Definite loop:** A loop that executes a finite number of times.
  - **Indefinite loop:** A loop where it is difficult to determine in advance how many times it will be executed.

WHILE LOOP

- In Java, while loop is used to iterate a part of the program several times.
- Here, condition is evaluated first, and if it returns true, the statements or lines inside while loop are execute.
- If we don't know the number of iterations in advance then the best suitable loop is while loop.
- Syntax:

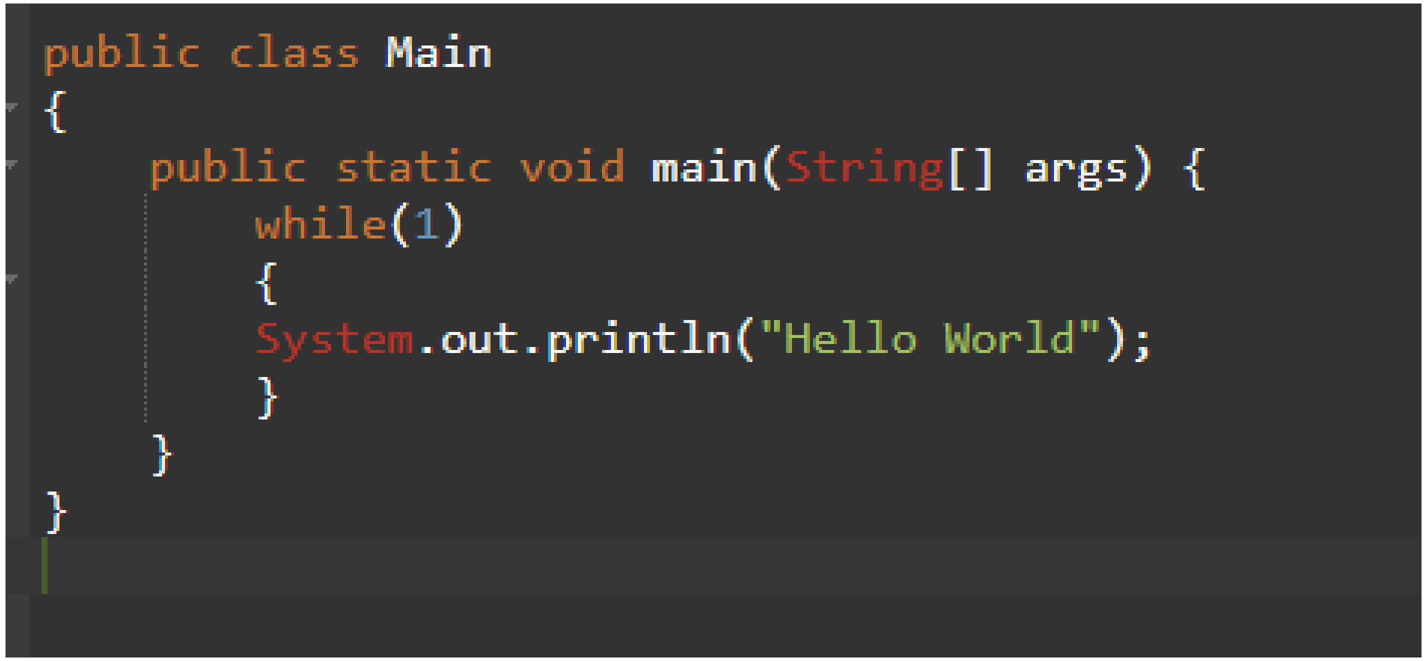
```
while(condition)
{
     Statement 1;
     Statement 2;
}
```

WHILE LOOP (CONT….)



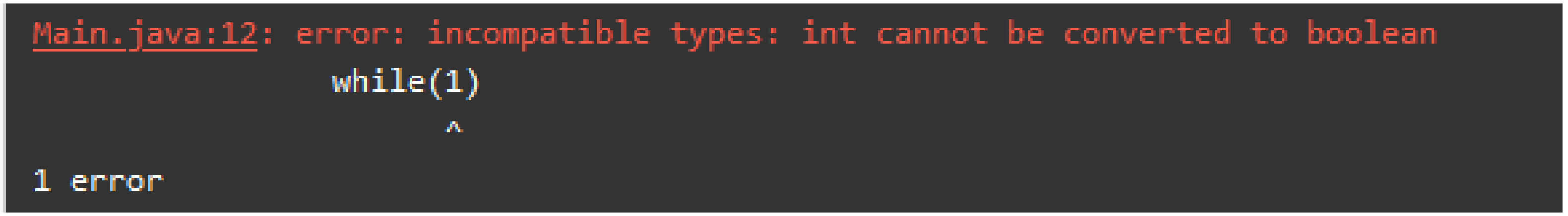
WHILE LOOP (CONT….)

- The argument to the while loop should be Boolean type. If we are using any other type we will get compile time error.



```
public class Main
{
    public static void main(String[] args) {
        while(1)
        {
        System.out.println("Hello World");
        }
    }
}
```

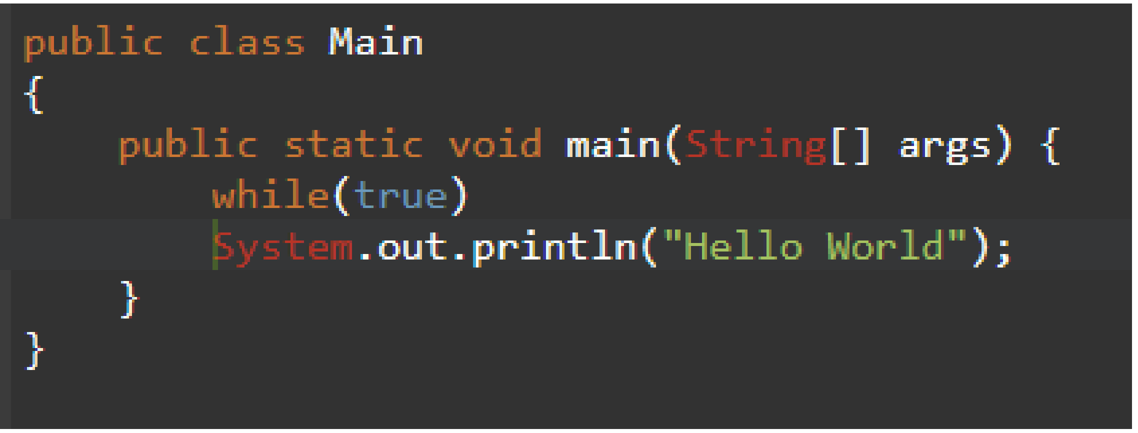
Output



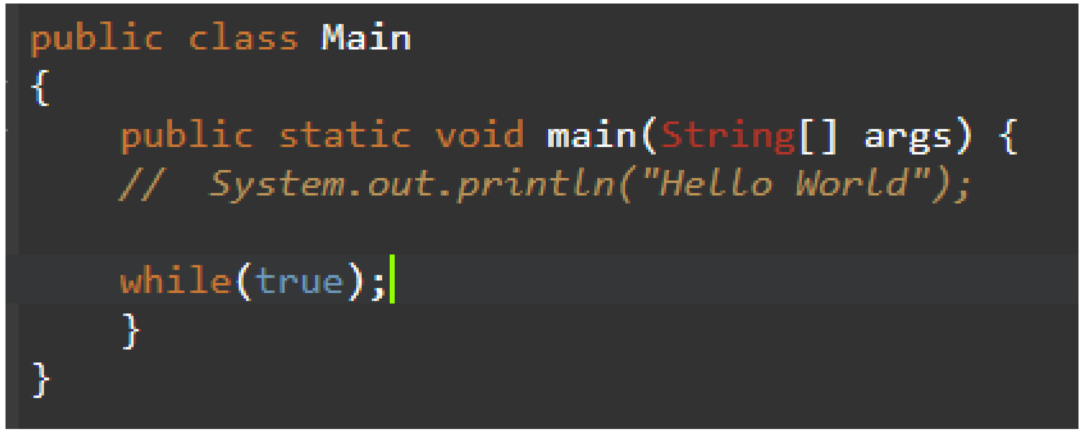
```
Main.java:12: error: incompatible types: int cannot be converted to boolean
                while(1)
                      ^
1 error
```

WHILE LOOP (CONT….)

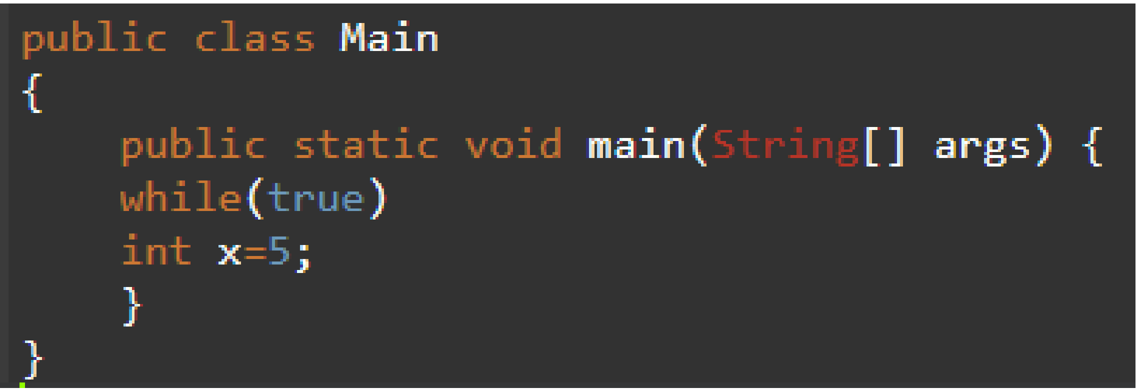
- Curly braces are optional and without curly braces we can take only one statement which should not be declarative statement.



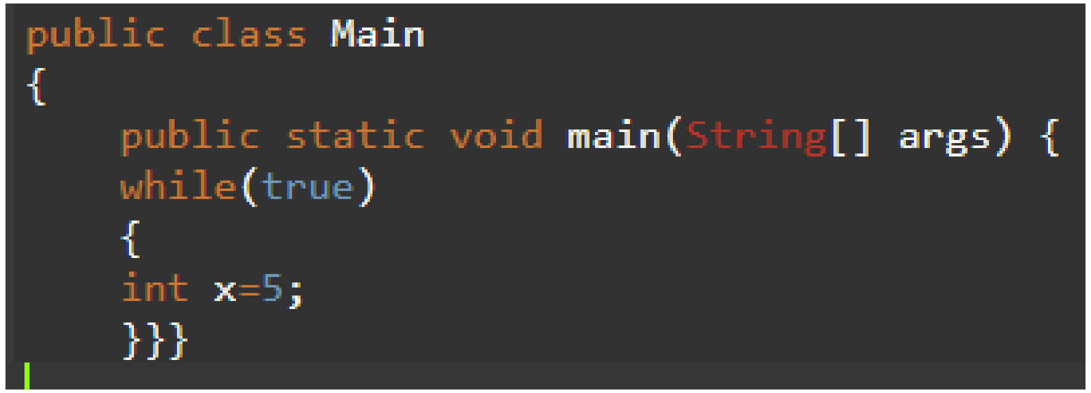
```
public class Main
{
    public static void main(String[] args) {
        while(true)
        System.out.println("Hello World");
    }
}
```



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

    while(true);
    }
}
```



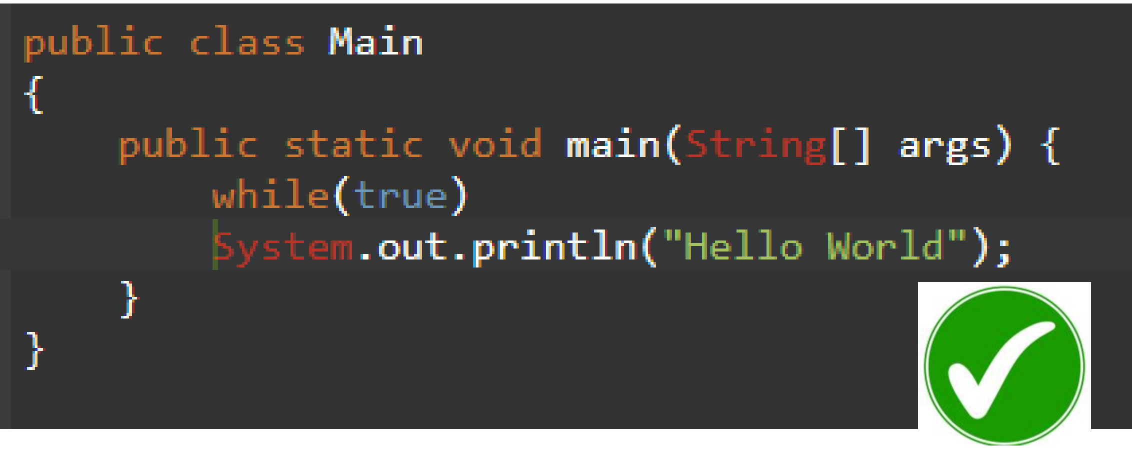
```
public class Main
{
    public static void main(String[] args) {
    while(true)
    int x=5;
    }
}
```



```
public class Main
{
    public static void main(String[] args) {
    while(true)
    {
    int x=5;
    }}}
```

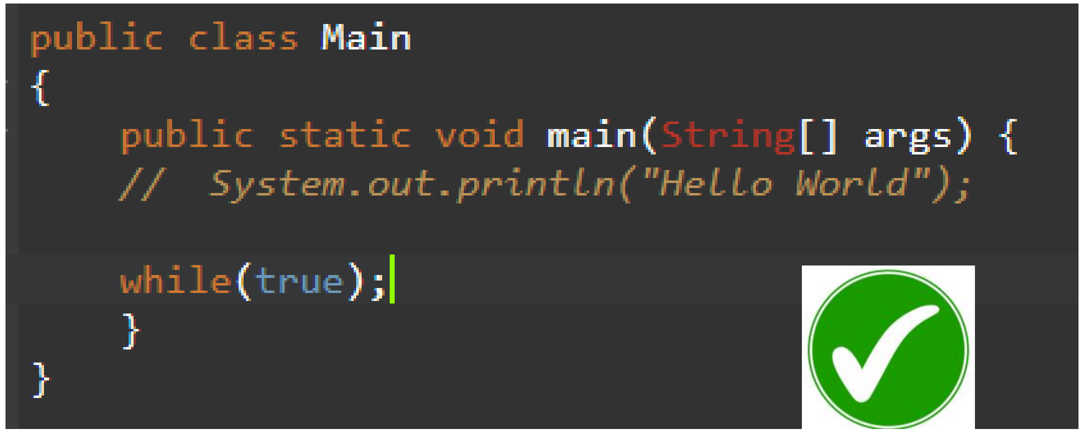
WHILE LOOP (CONT….)

- Curly braces are optional and without curly braces we can take only one statement which should not be declarative statement.



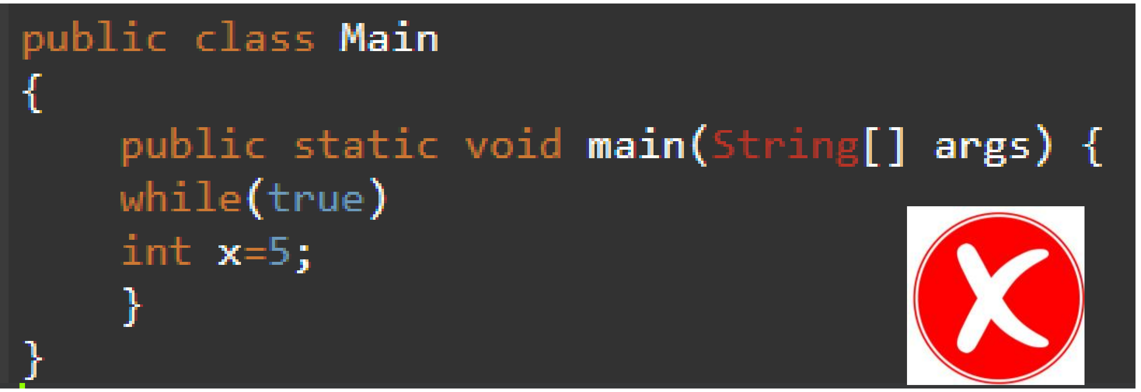
```
public class Main
{
    public static void main(String[] args) {
        while(true)
        System.out.println("Hello World");
    }
}
```

✓



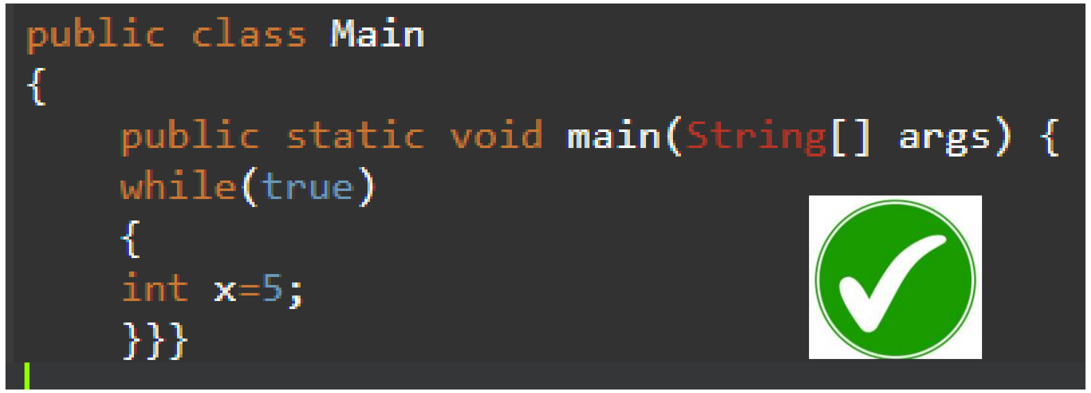
```
public class Main
{
    public static void main(String[] args) {
    //  System.out.println("Hello World");

    while(true);
    }
}
```

✓



```
public class Main
{
    public static void main(String[] args) {
    while(true)
    int x=5;
    }
}
```

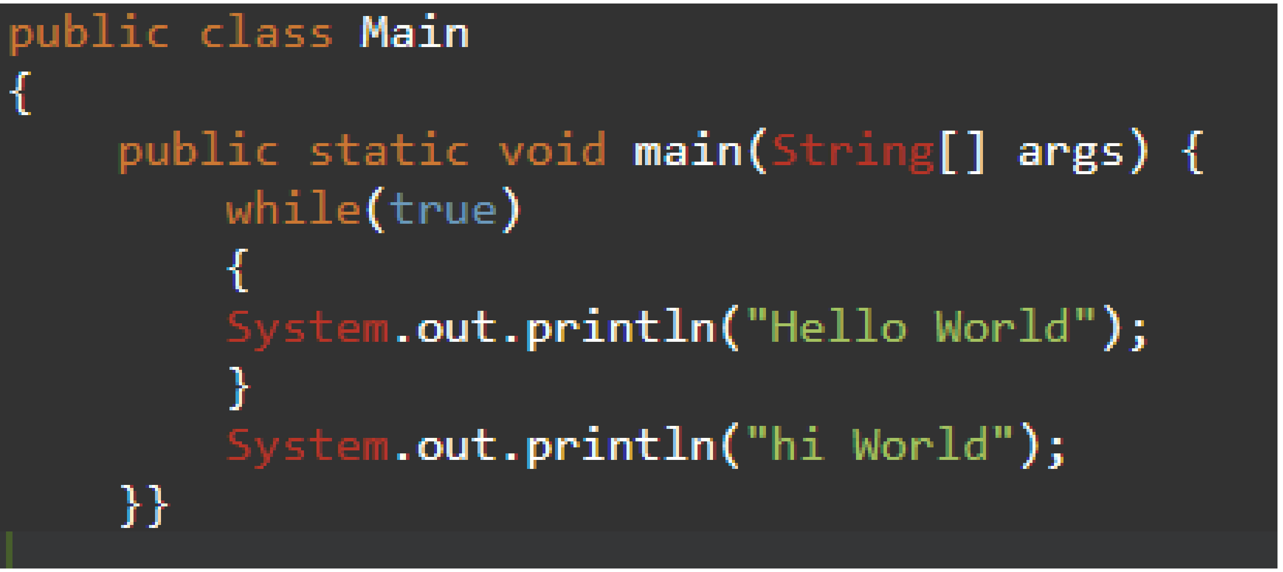
✗



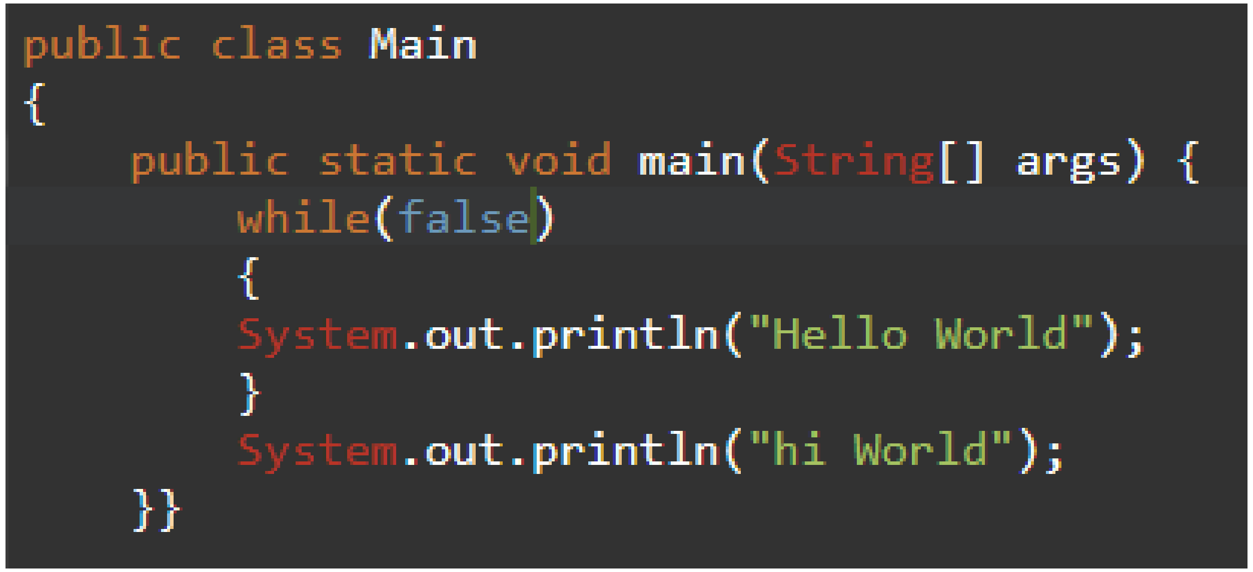
```
public class Main
{
    public static void main(String[] args) {
    while(true)
    {
    int x=5;
    }}}
```

✓

WHILE LOOP (CONT….)

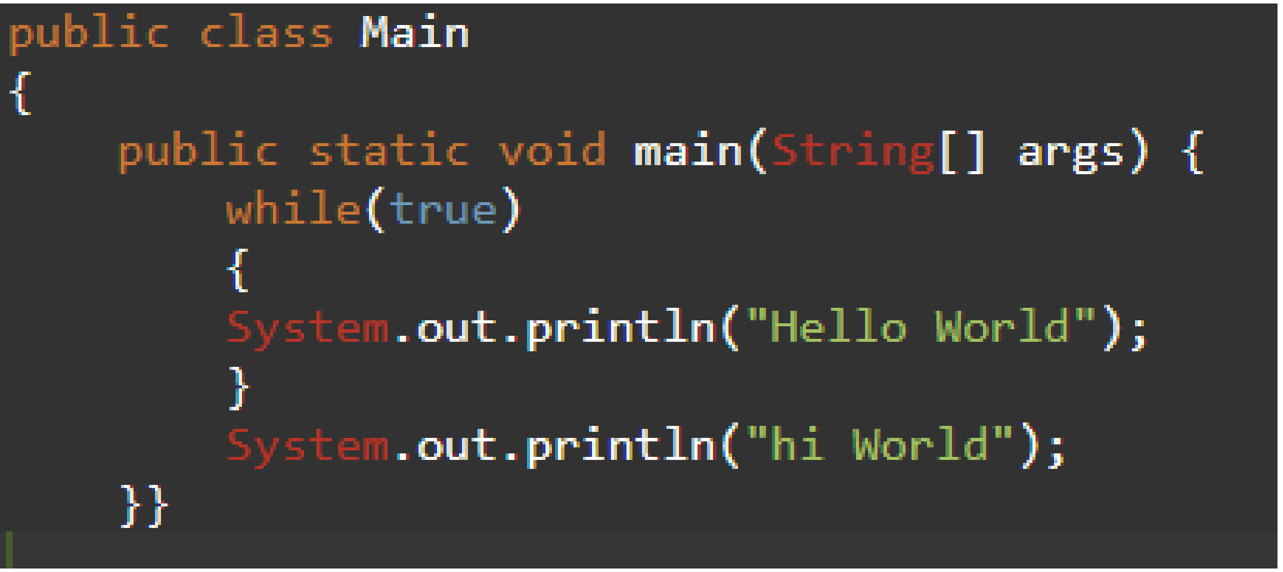


```
public class Main
{
    public static void main(String[] args) {
        while(true)
        {
        System.out.println("Hello World");
        }
        System.out.println("hi World");
    }}
```

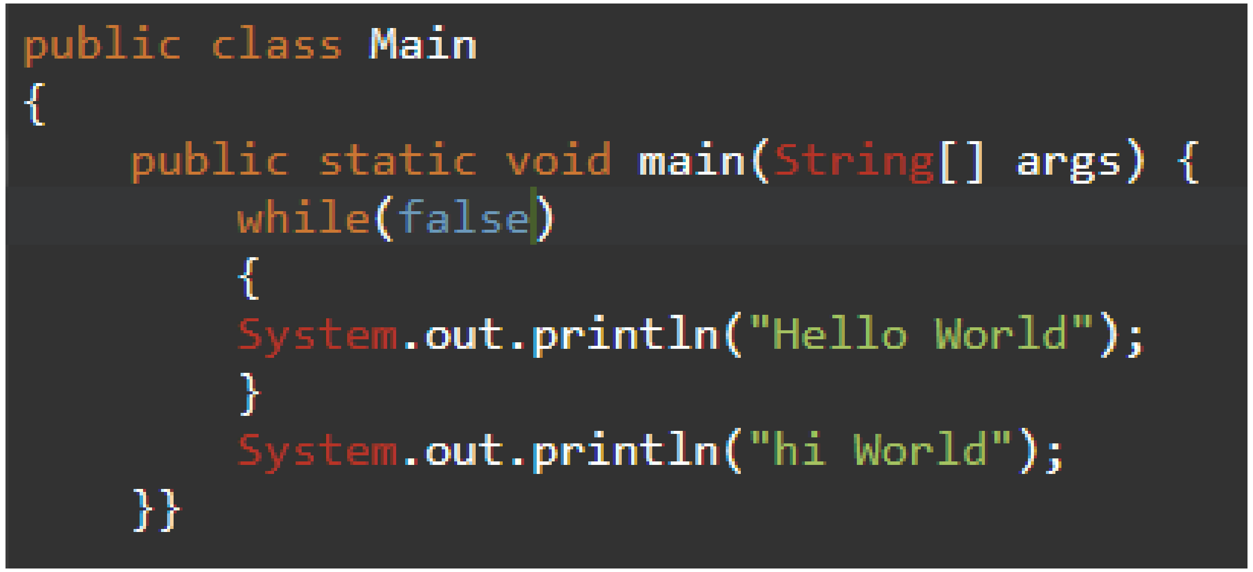


```
public class Main
{
    public static void main(String[] args) {
        while(false)
        {
        System.out.println("Hello World");
        }
        System.out.println("hi World");
    }}
```

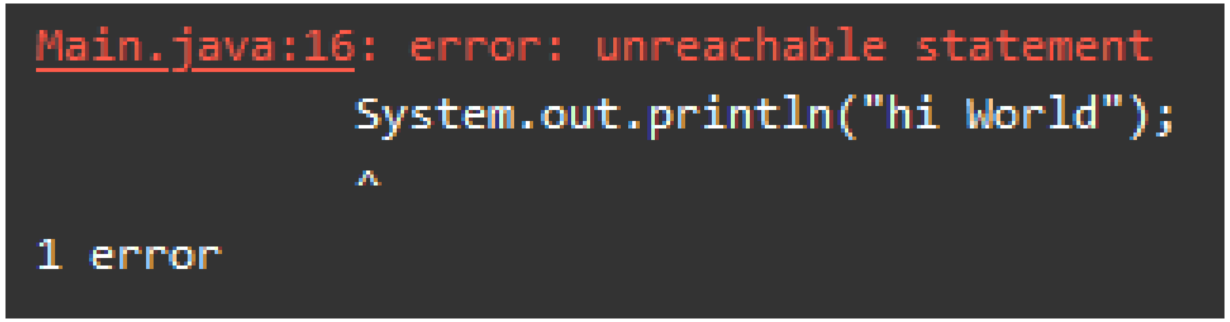
WHILE LOOP (CONT….)



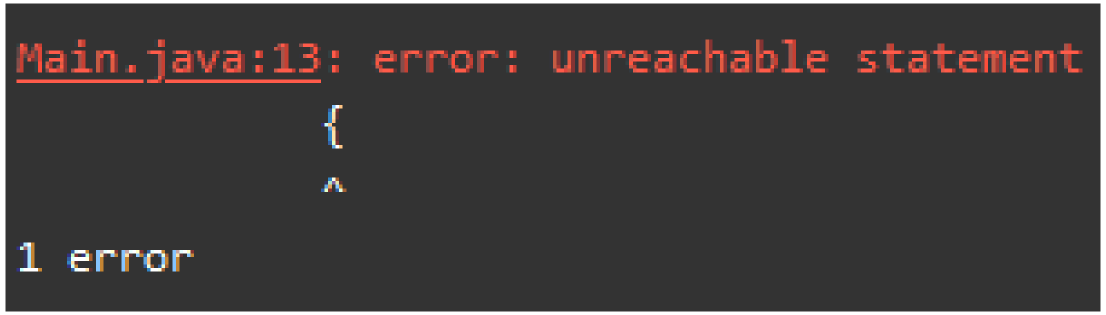
```java
public class Main
{
    public static void main(String[] args) {
        while(true)
        {
        System.out.println("Hello World");
        }
        System.out.println("hi World");
    }}
```



```java
public class Main
{
    public static void main(String[] args) {
        while(false)
        {
        System.out.println("Hello World");
        }
        System.out.println("hi World");
    }}
```

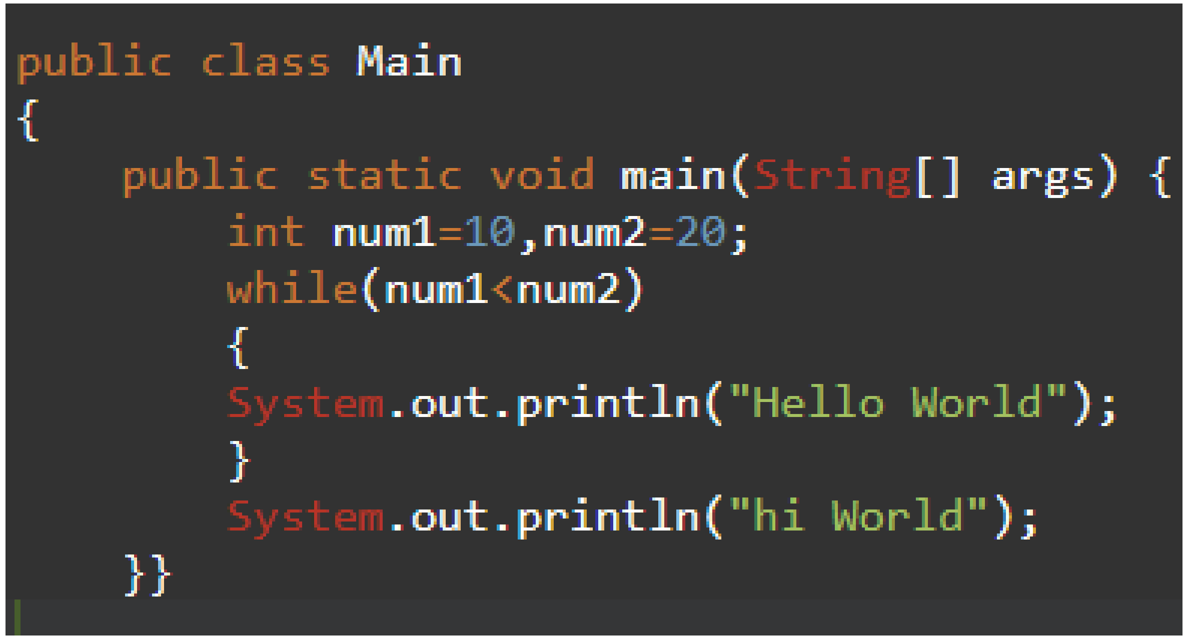


```
Main.java:16: error: unreachable statement
          System.out.println("hi World");
          ^
1 error
```

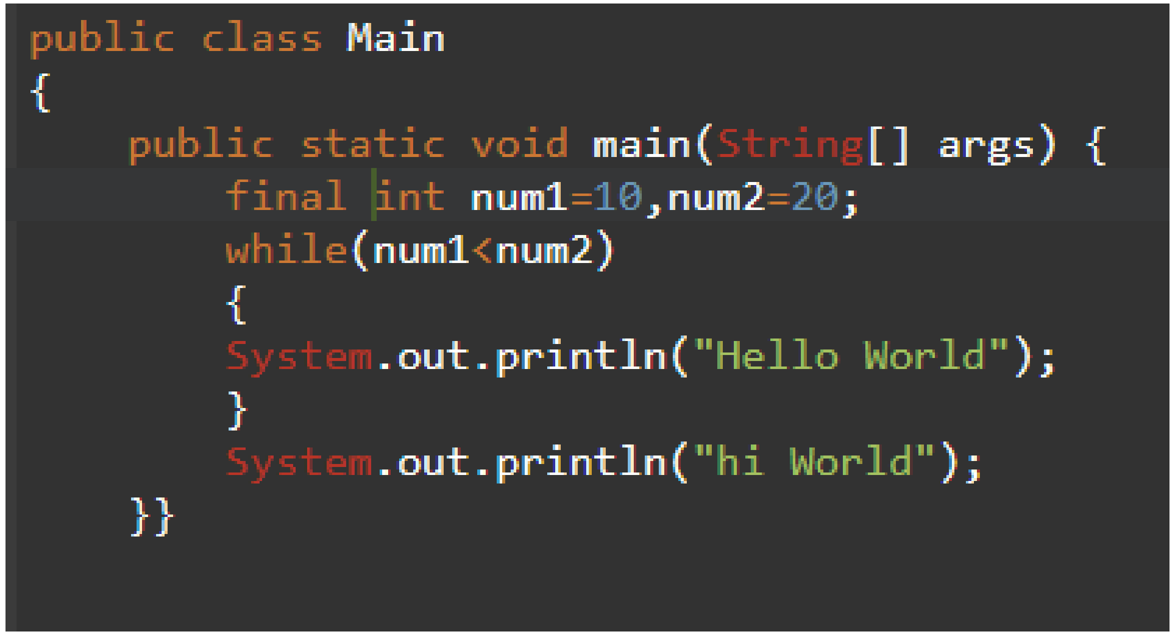


```
Main.java:13: error: unreachable statement
          {
          ^
1 error
```

WHILE LOOP (CONT….)

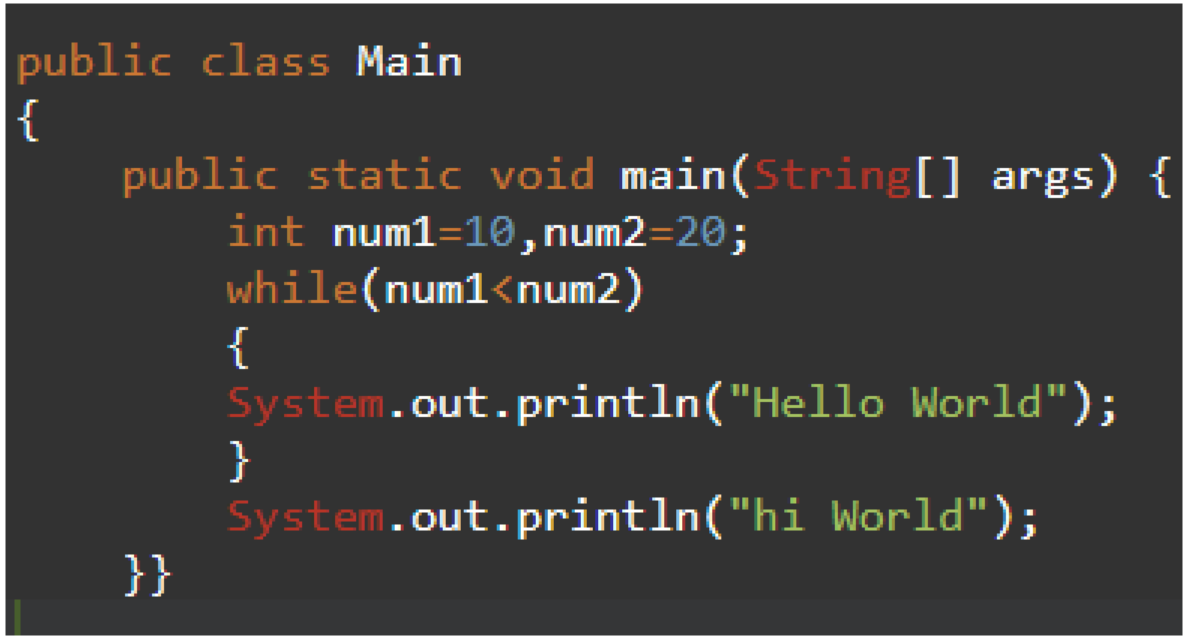


```
public class Main
{
    public static void main(String[] args) {
        int num1=10,num2=20;
        while(num1<num2)
        {
        System.out.println("Hello World");
        }
        System.out.println("hi World");
    }}
```

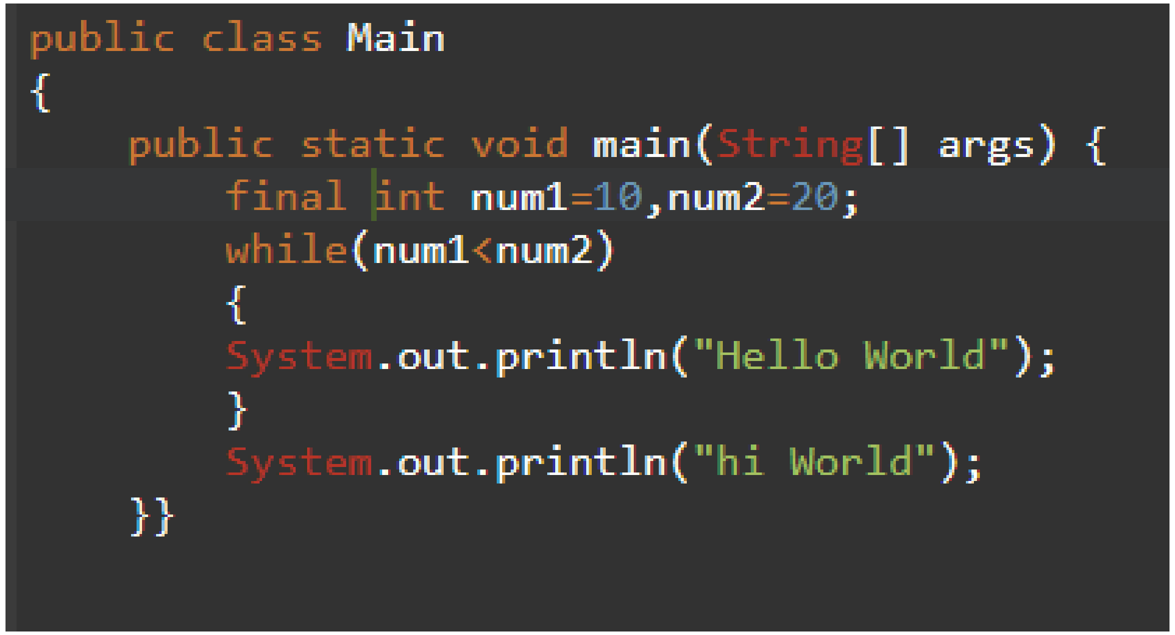


```
public class Main
{
    public static void main(String[] args) {
        final int num1=10,num2=20;
        while(num1<num2)
        {
        System.out.println("Hello World");
        }
        System.out.println("hi World");
    }}
```

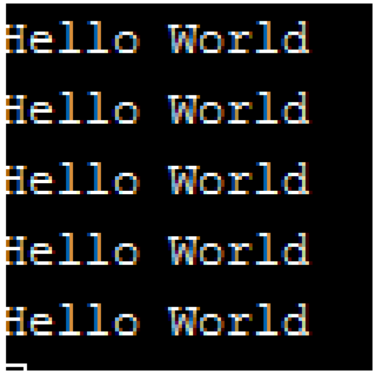
WHILE LOOP (CONT….)



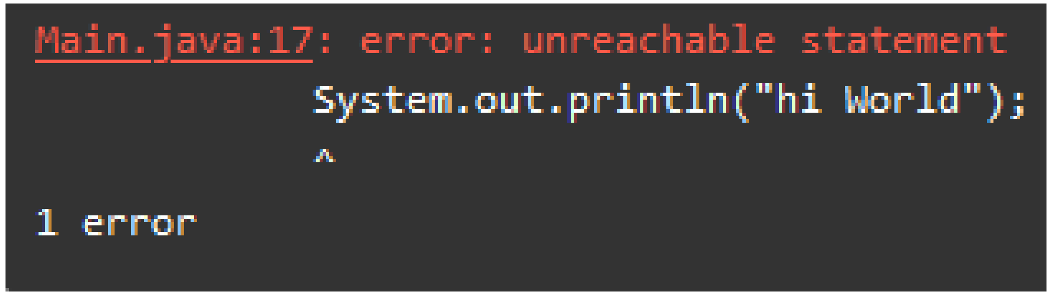
```
public class Main
{
    public static void main(String[] args) {
        int num1=10,num2=20;
        while(num1<num2)
        {
        System.out.println("Hello World");
        }
        System.out.println("hi World");
    }}
```



```
public class Main
{
    public static void main(String[] args) {
        final int num1=10,num2=20;
        while(num1<num2)
        {
        System.out.println("Hello World");
        }
        System.out.println("hi World");
    }}
```

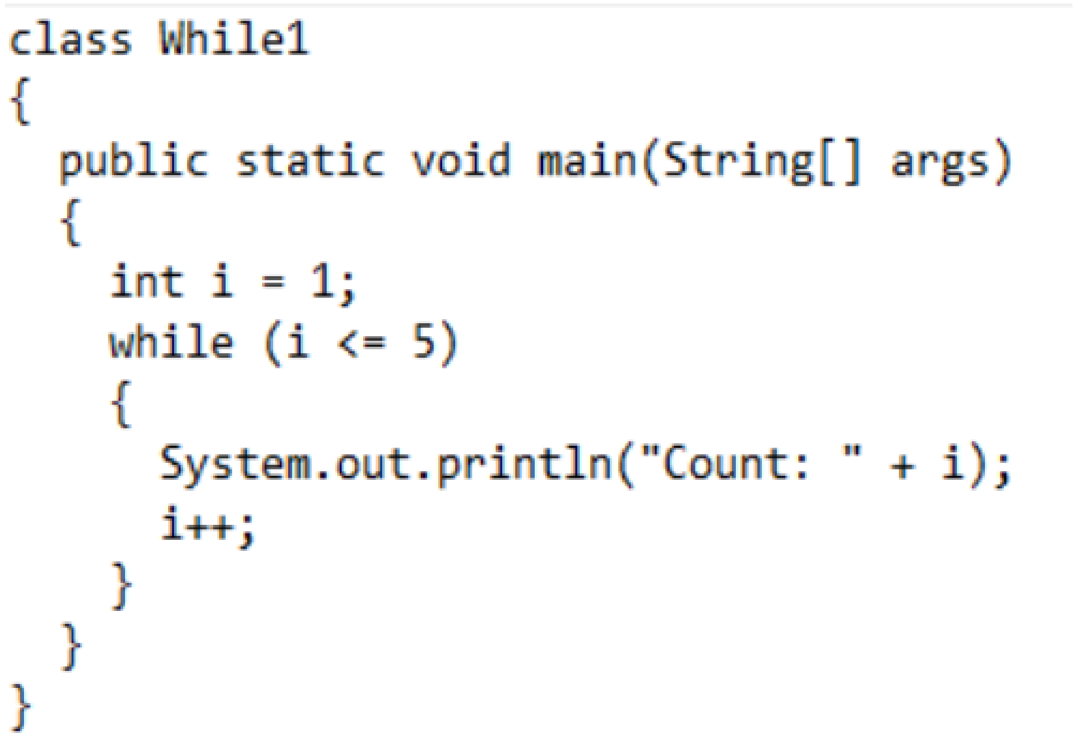


```
Hello World
Hello World
Hello World
Hello World
Hello World
```



```
Main.java:17: error: unreachable statement
          System.out.println("hi World");
          ^
1 error
```

SIMPLE PROGRAM USING WHILE LOOP



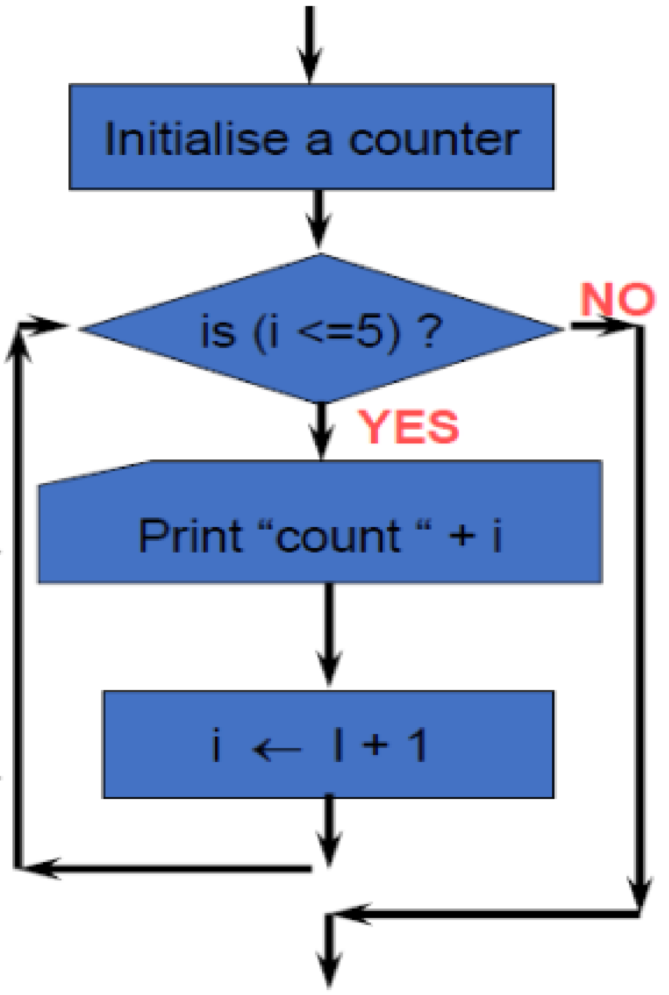
```
class While1
{
  public static void main(String[] args)
  {
    int i = 1;
    while (i <= 5)
    {
      System.out.println("Count: " + i);
      i++;
    }
  }
}
```

Output:

```
Count: 1
Count: 2
Count: 3
Count: 4
Count: 5
```

SIMPLE PROGRAM USING WHILE LOOP (CONT..)

```
int i = 1;

while (i <= 5)
{

System.out.println ("count : " + i);

    i++;
}
```

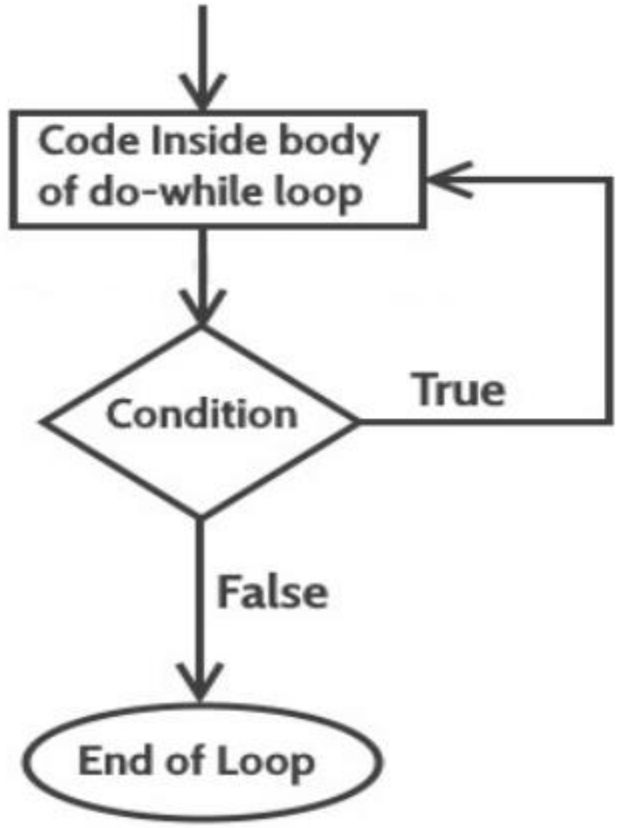


DO-WHILE LOOP

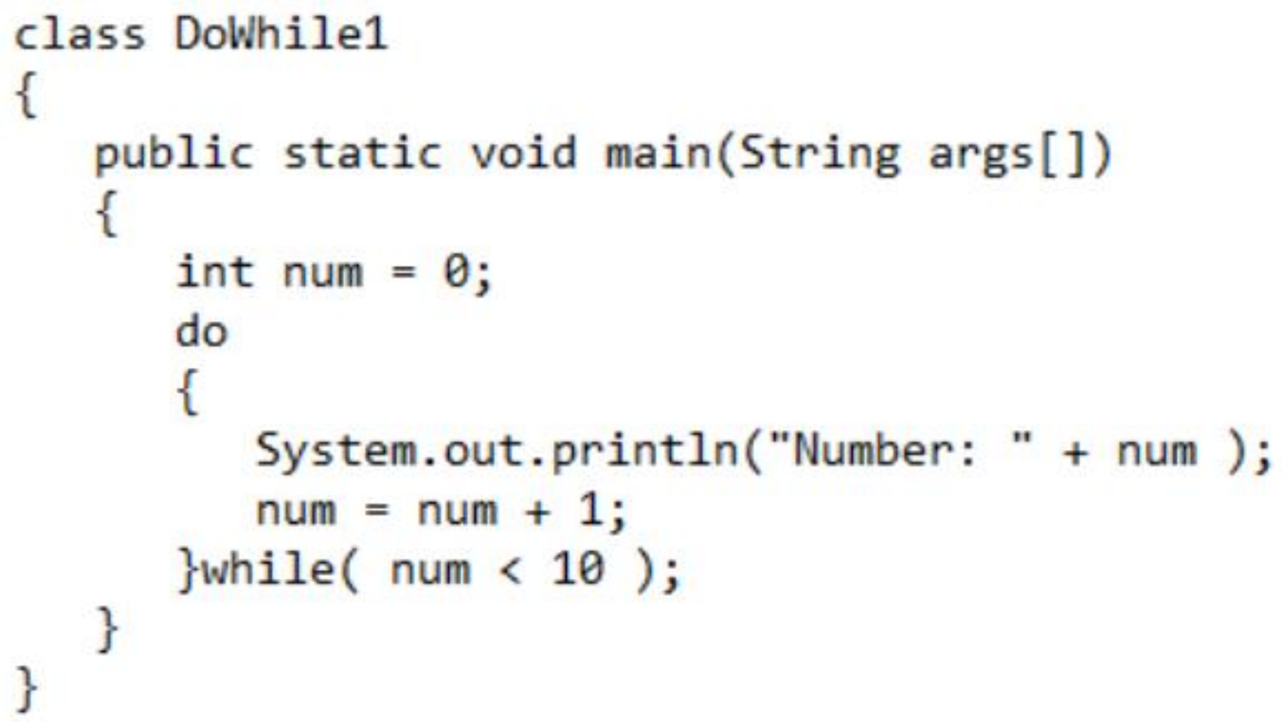
- Do-while loop is actually one kind of while loop.
- This loop executes the lines or statements once before checking the condition. If the condition is true, it repeats the loop as long as the given condition is true.
- If we want to execute loop body at least once then we should go for do-while loop
- **Syntax:**

```
do
{
     Statement 1;
    Statement 2 ;
}
while(condition);
```

DO-WHILE LOOP(CONT….)



DO-WHILE LOOP(CONT….)



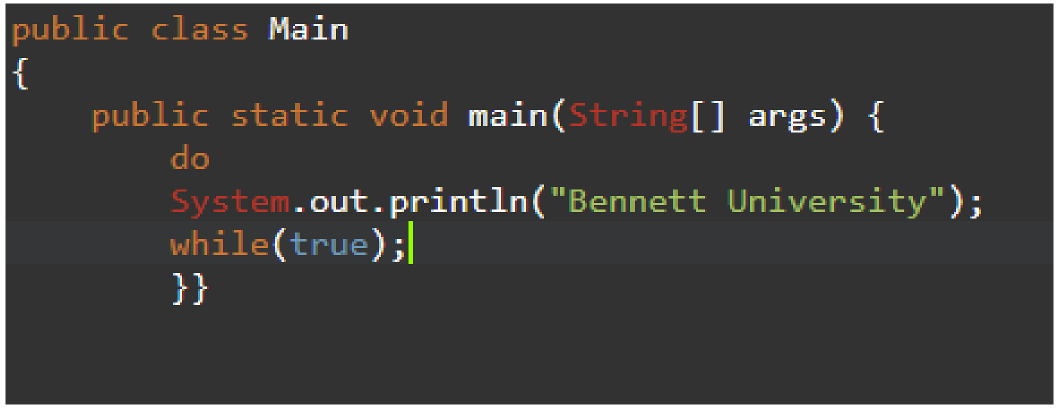
```
class DoWhile1
{
    public static void main(String args[])
    {
        int num = 0;
        do
        {
            System.out.println("Number: " + num );
            num = num + 1;
        }while( num < 10 );
    }
}
```

Output:

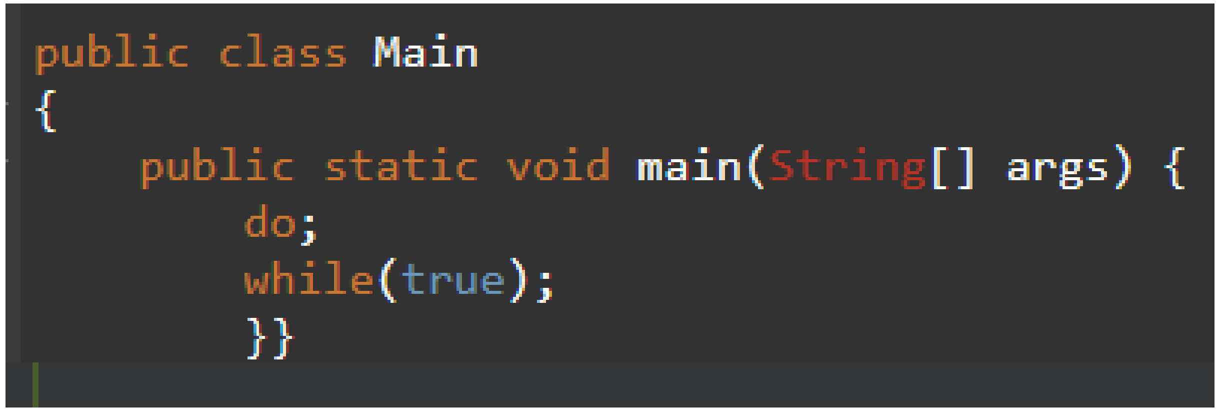
```
      Number: 0
      Number: 1
      Number: 2
      Number: 3
      Number: 4
      Number: 5
      Number: 6
      Number: 7
      Number: 8
      Number: 9
```

DO-WHILE LOOP

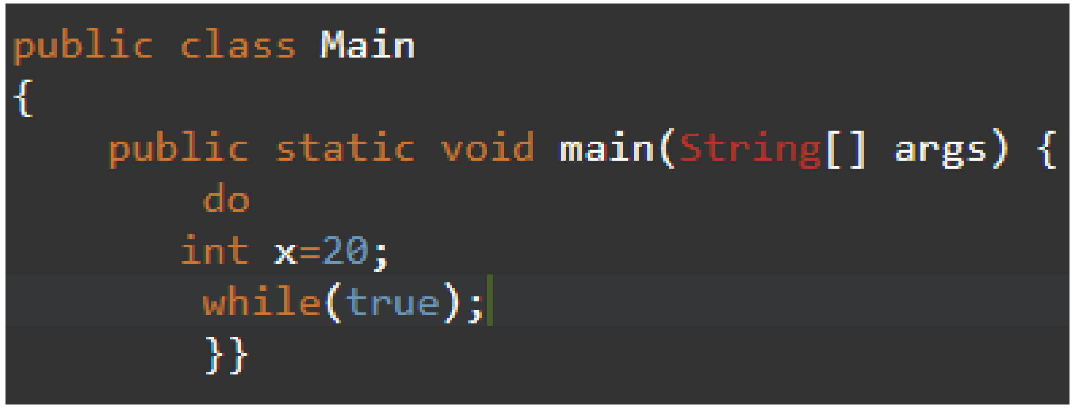
- Curly braces are optional and without having curly braces we can take only one statement between do-while and should not be declarative statement.



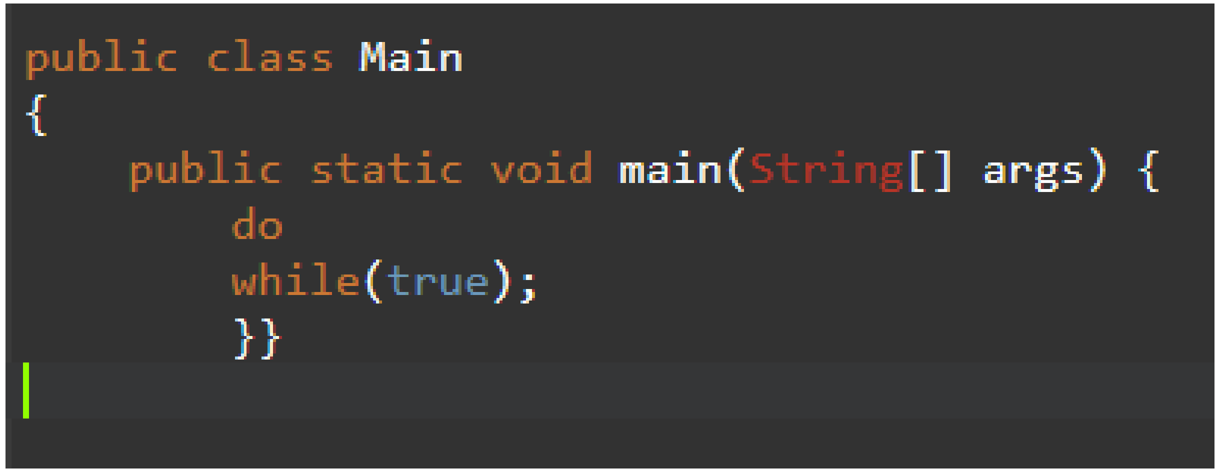
```
public class Main
{
    public static void main(String[] args) {
        do
        System.out.println("Bennett University");
        while(true);
        }}
```



```
public class Main
{
    public static void main(String[] args) {
        do;
        while(true);
        }}
```



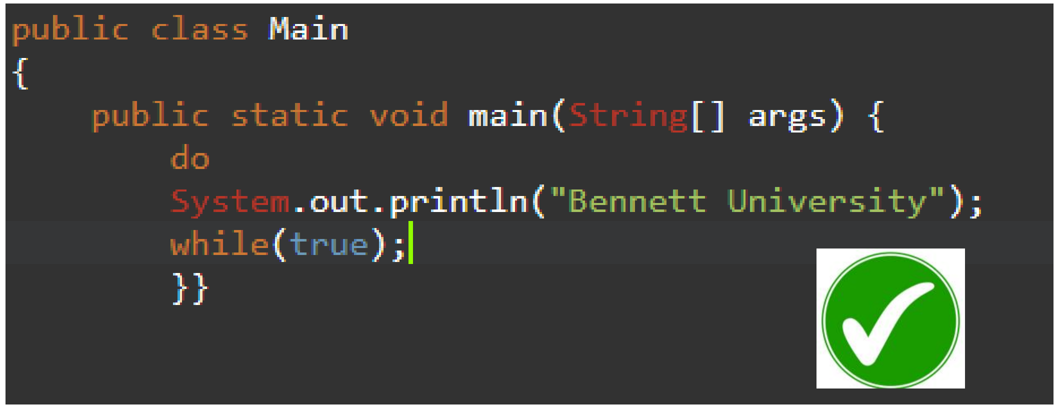
```
public class Main
{
    public static void main(String[] args) {
        do
       int x=20;
        while(true);
        }}
```



```
public class Main
{
    public static void main(String[] args) {
        do
        while(true);
        }}
```

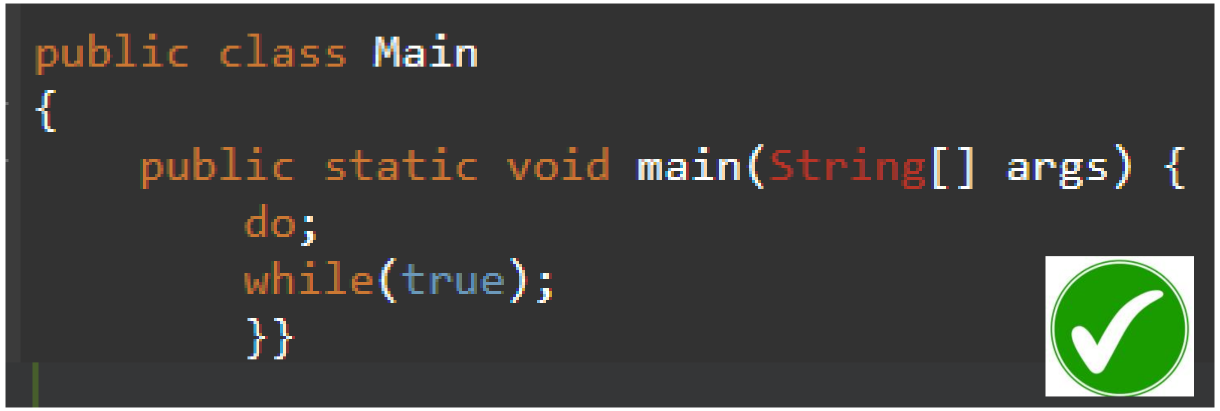
DO-WHILE LOOP

- Curly braces are optional and without having curly braces we can take only one statement between do-while and should not be declarative statement.



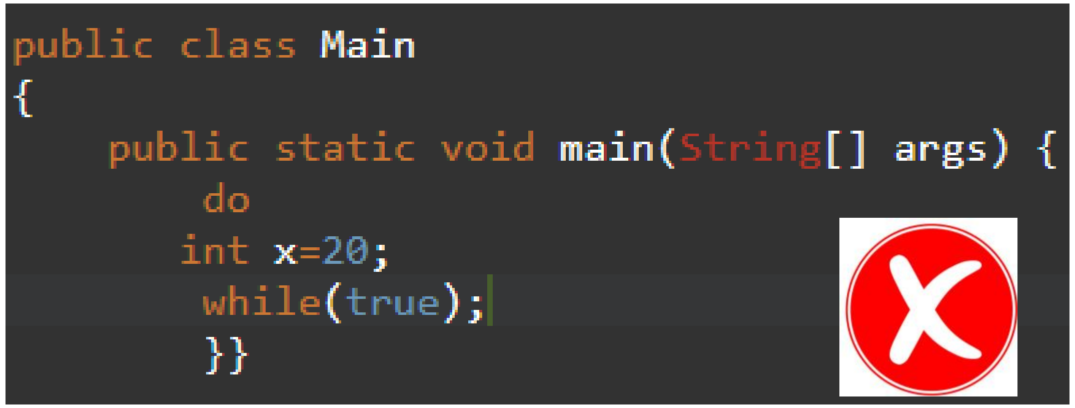
```
public class Main
{
    public static void main(String[] args) {
        do
        System.out.println("Bennett University");
        while(true);
        }}
```

✓



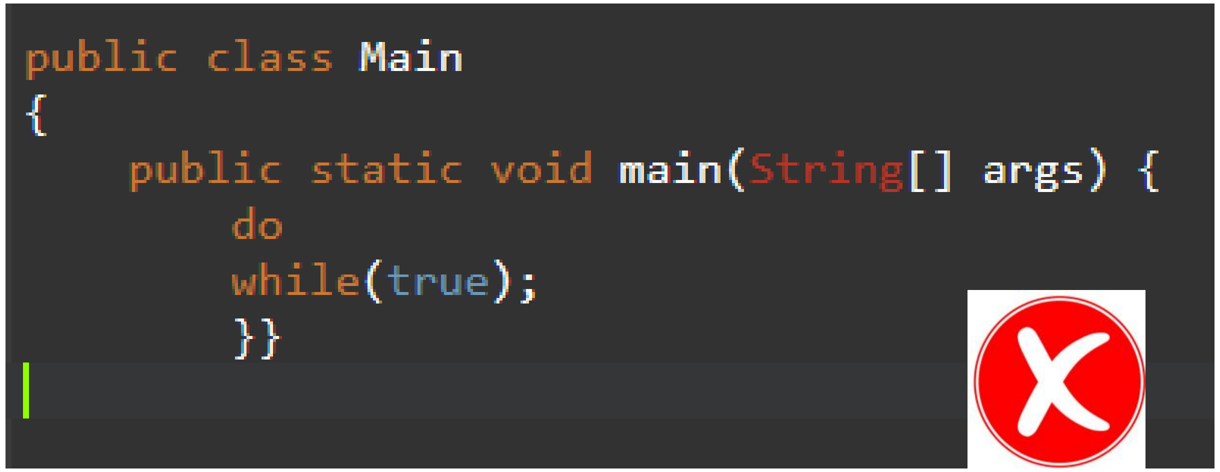
```
public class Main
{
    public static void main(String[] args) {
        do;
        while(true);
        }}
```

✓



```
public class Main
{
    public static void main(String[] args) {
        do
       int x=20;
        while(true);
        }}
```

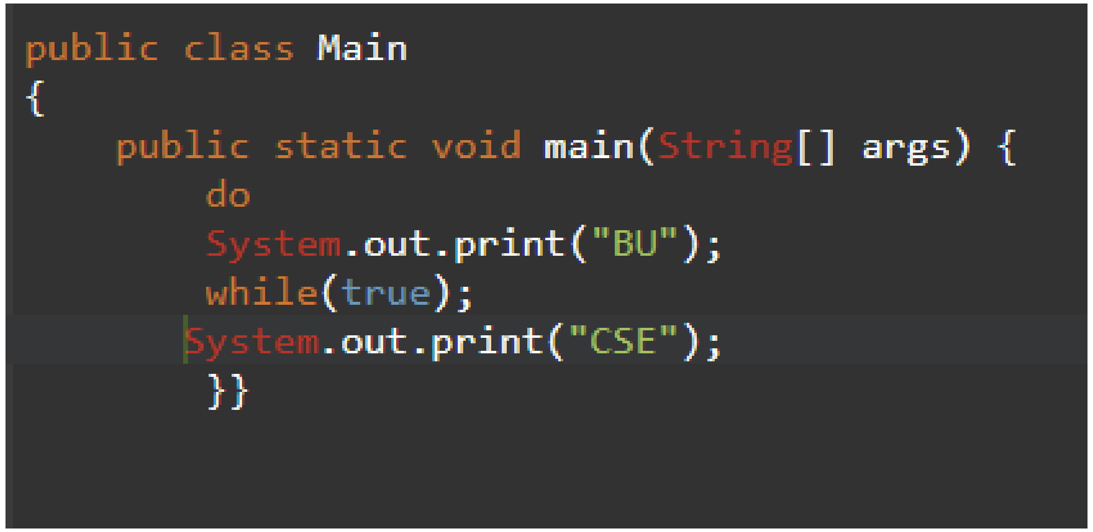
✗



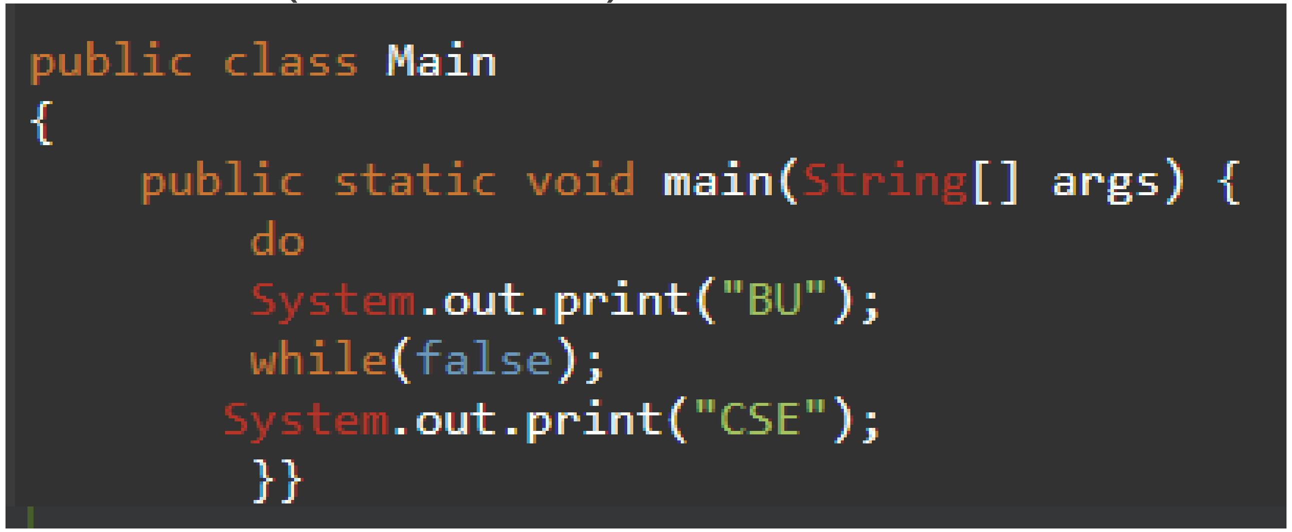
```
public class Main
{
    public static void main(String[] args) {
        do
        while(true);
        }}
```

✗

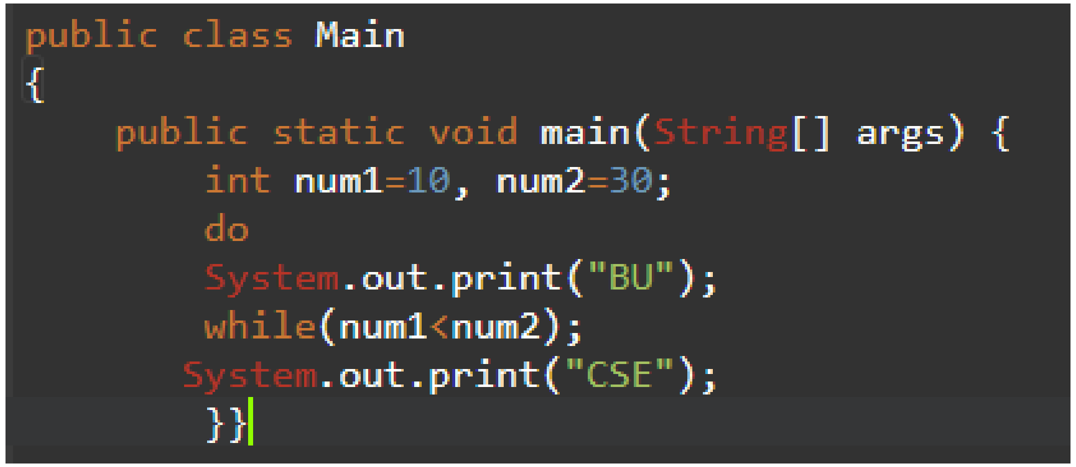
DO-WHILE LOOP (EXAMPLE)



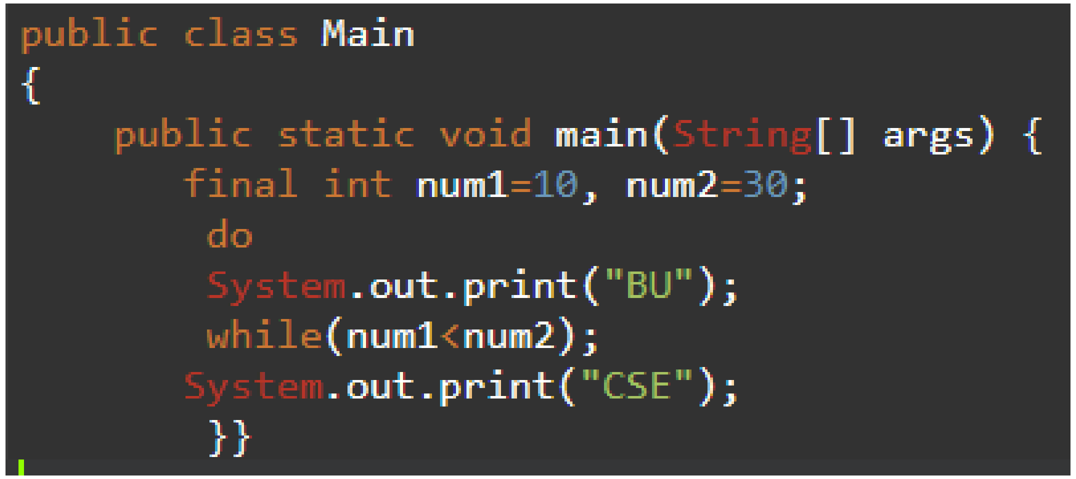
```
public class Main
{
    public static void main(String[] args) {
        do
        System.out.print("BU");
        while(true);
       System.out.print("CSE");
        }}
```



```
public class Main
{
    public static void main(String[] args) {
        do
        System.out.print("BU");
        while(false);
       System.out.print("CSE");
        }}
```

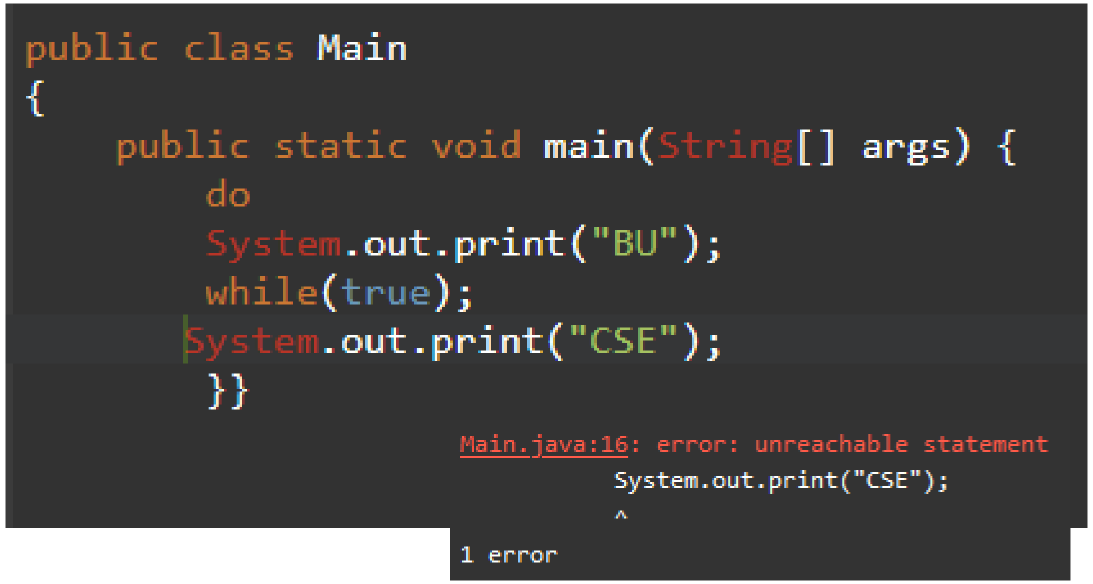


```
public class Main
{
    public static void main(String[] args) {
        int num1=10, num2=30;
        do
        System.out.print("BU");
        while(num1<num2);
       System.out.print("CSE");
        }}
```



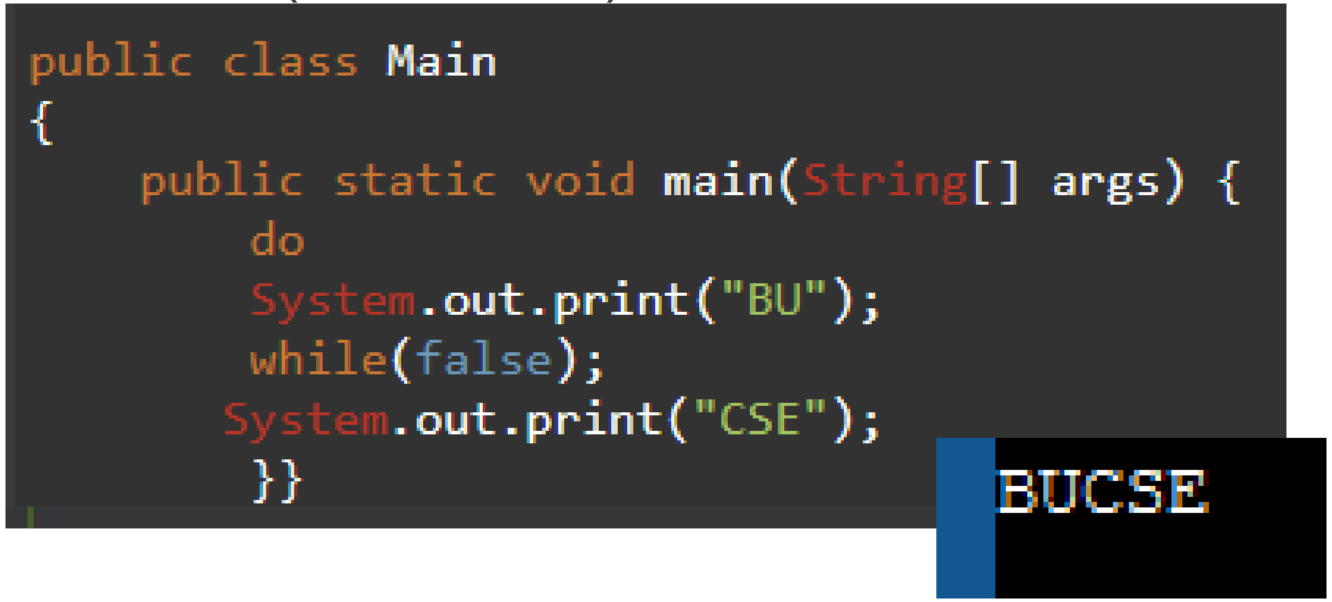
```
public class Main
{
    public static void main(String[] args) {
        final int num1=10, num2=30;
         do
         System.out.print("BU");
         while(num1<num2);
        System.out.print("CSE");
         }}
```

DO-WHILE LOOP (EXAMPLE)



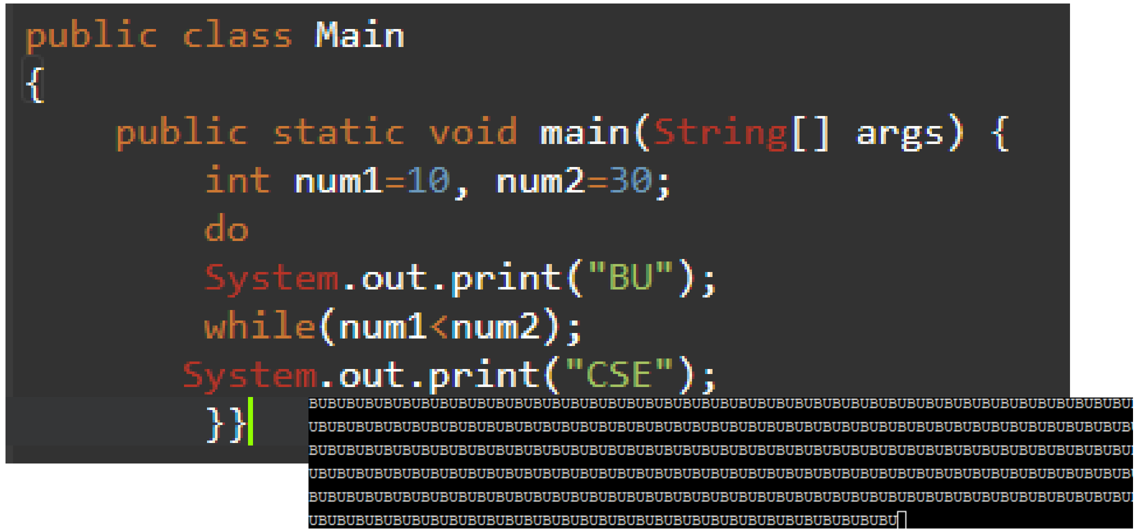
```
public class Main
{
    public static void main(String[] args) {
        do
        System.out.print("BU");
        while(true);
       System.out.print("CSE");
        }}
```

```
Main.java:16: error: unreachable statement
        System.out.print("CSE");
        ^
1 error
```



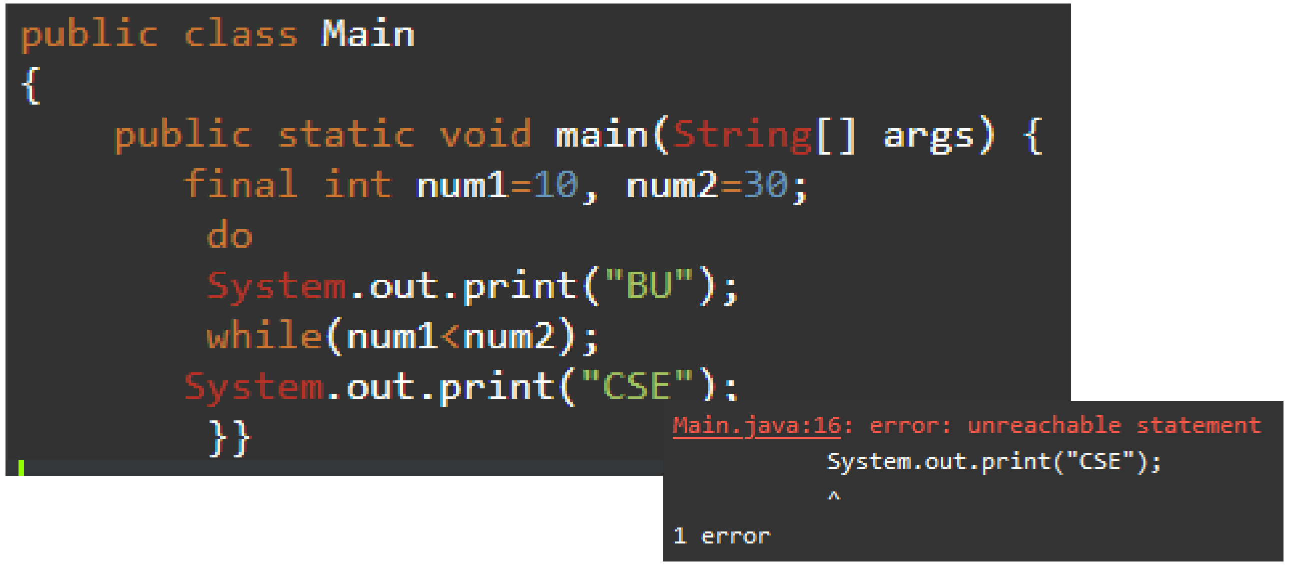
```
public class Main
{
    public static void main(String[] args) {
        do
        System.out.print("BU");
        while(false);
       System.out.print("CSE");
        }}
```

```
BUCSE
```



```
public class Main
{
    public static void main(String[] args) {
        int num1=10, num2=30;
        do
        System.out.print("BU");
        while(num1<num2);
       System.out.print("CSE");
        }}
```

```
BUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBU
UBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUB
BUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBU
UBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUB
BUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBU
UBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBUBU
```



```
public class Main
{
    public static void main(String[] args) {
      final int num1=10, num2=30;
       do
       System.out.print("BU");
       while(num1<num2);
      System.out.print("CSE");
       }}
```

```
Main.java:16: error: unreachable statement
        System.out.print("CSE");
        ^
1 error
```

FOR LOOP

- ❑ For loop is concise version of while loop.
- ❑ For loop is used, when we know exactly for how many times the code block will be executed.
- ❑ There are mainly four parts in the entire for loop:

  - ❖**Initialization**
  - ❖**Condition**
  - ❖**Increment/decrement**
  - ❖**Statement**

FOR LOOP (CONT..)

- **Syntax:**

```
for(initialization; condition (Boolean Expression); increment/decrement)
{
    statement 1;
    statement 2;
}
```

Note: Curly braces are optional & without curly braces we can take only one statement which should not be declarative

Initialization Section:

- **This will be executed only once.**
- **Usually we declaring and performing initialization for the variables in this section.**
- **Here we can declare multiple variables of same datatype but different datatype variables we can’t declare.**
- **Example: int i=0, j=1;**
- **int i=0, byte b=2;**
- **int i=0, int j=0;**

FOR LOOP (CONT..)

❑ **Syntax:**

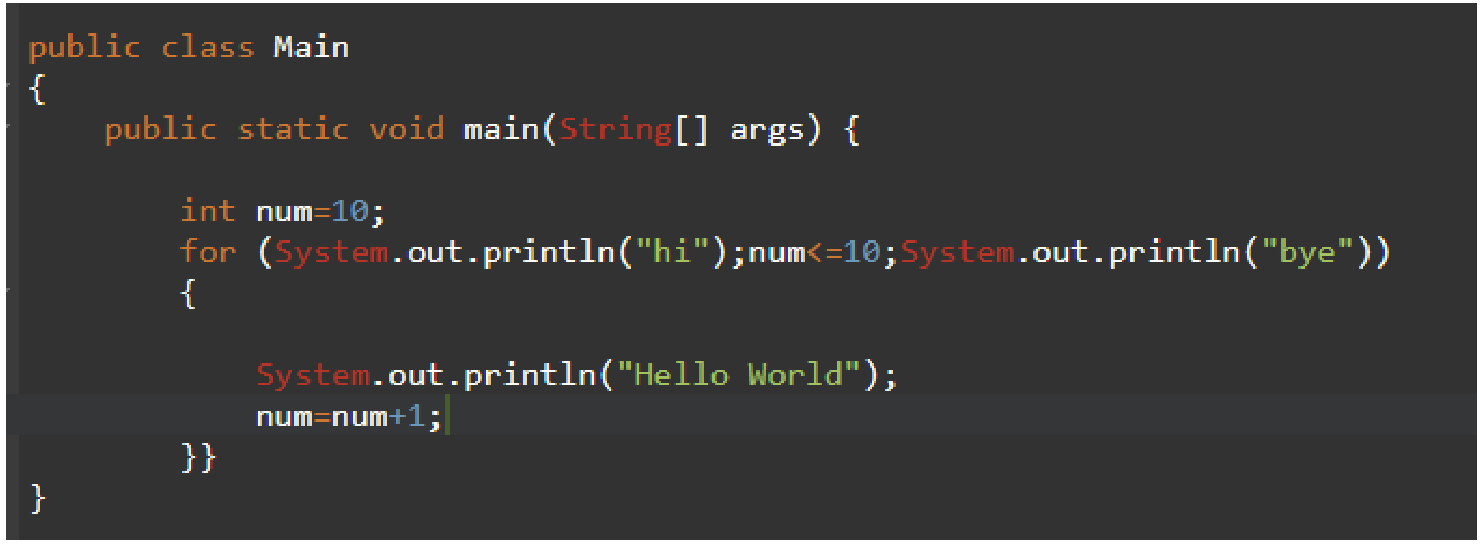
```
for(initialization; condition (Boolean Expression); increment/decrement)
{
    statement 1;
    statement 2;
}
```

Note: Curly braces are optional & without curly braces we can take only one statement which should not be declarative

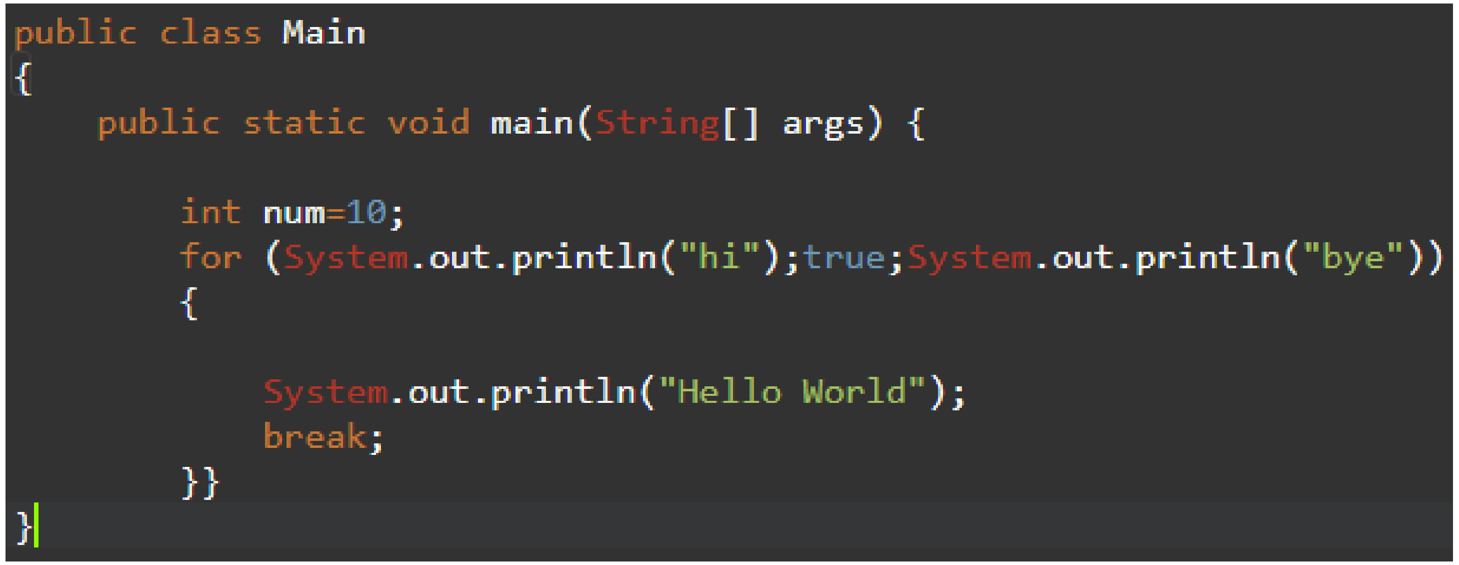
Initialization Section:

- **This will be executed only once.**
- **Usually we declaring and performing initialization for the variables in this section.**
- **Here we can declare multiple variables of same datatype but different datatype variables we can't declare.**
- **Example: int i=0, j=1;** ✓
- **int i=0, byte b=2;** ✗
- **int i=0, int j=0;** ✗
- **In the initialization section we can take any valid java statement including :System.out.print("") also._**

FOR LOOP (CONT..)

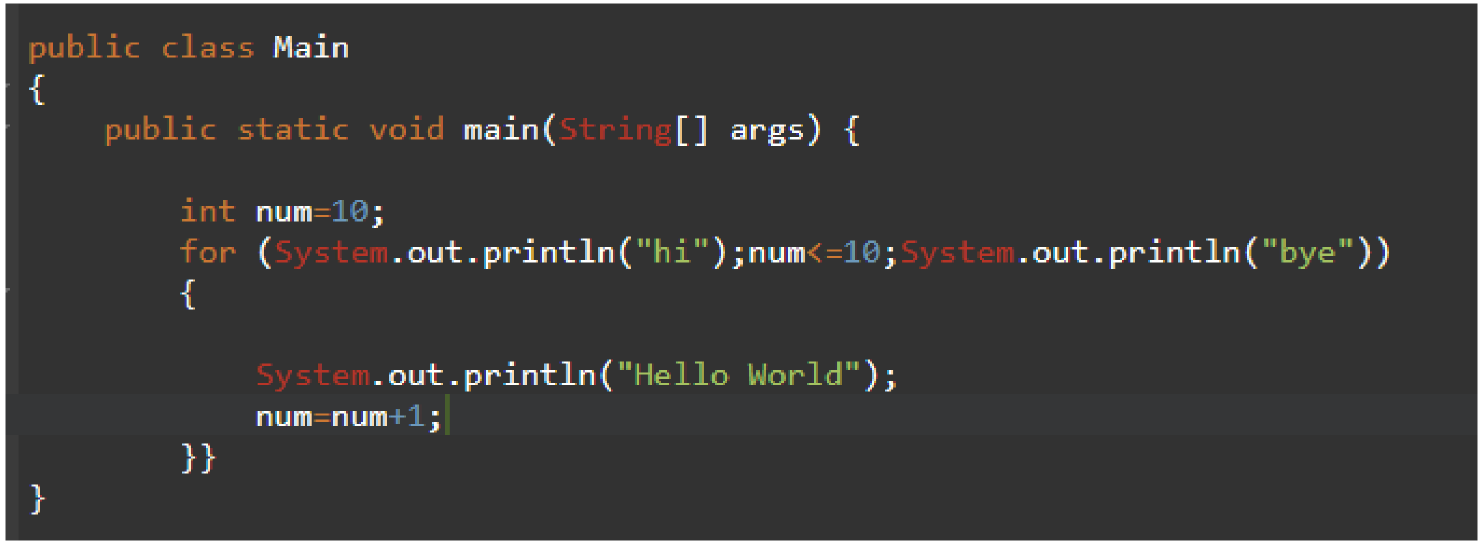


```
public class Main
{
    public static void main(String[] args) {

        int num=10;
        for (System.out.println("hi");num<=10;System.out.println("bye"))
        {

            System.out.println("Hello World");
            num=num+1;
        }}
}
```

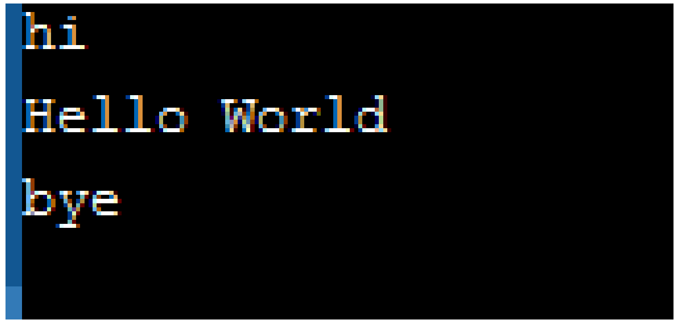


```
public class Main
{
    public static void main(String[] args) {

        int num=10;
        for (System.out.println("hi");true;System.out.println("bye"))
        {

            System.out.println("Hello World");
            break;
        }}
}
```

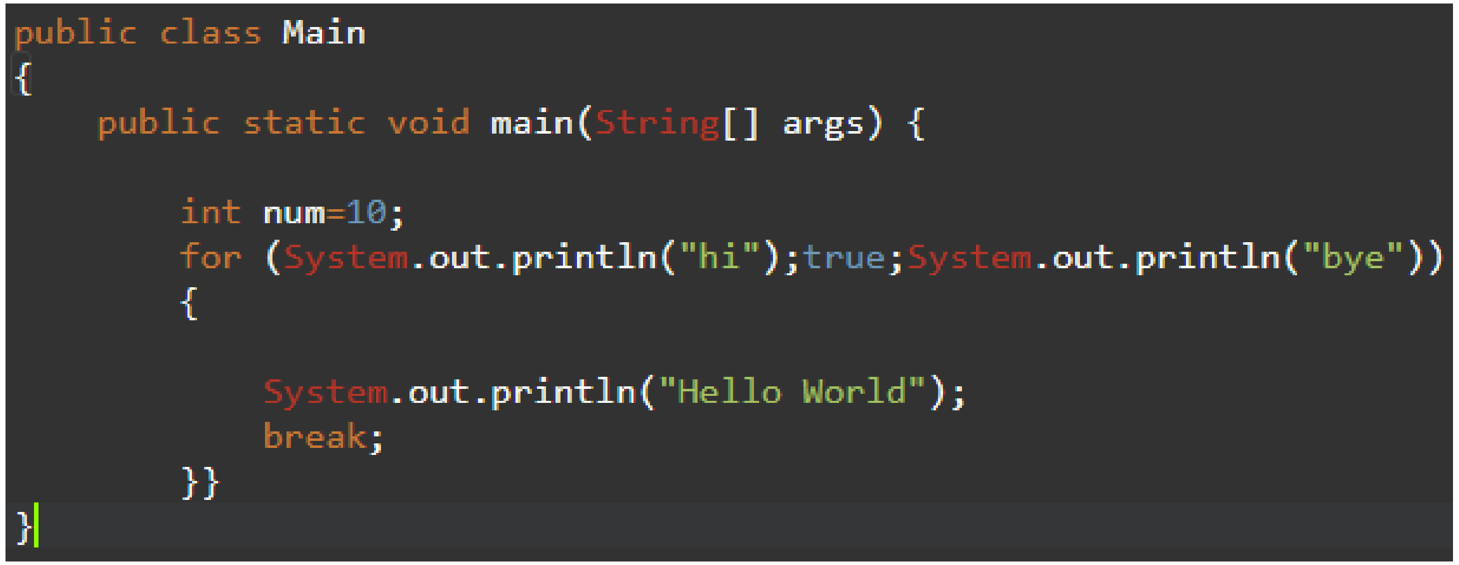
FOR LOOP (CONT..)



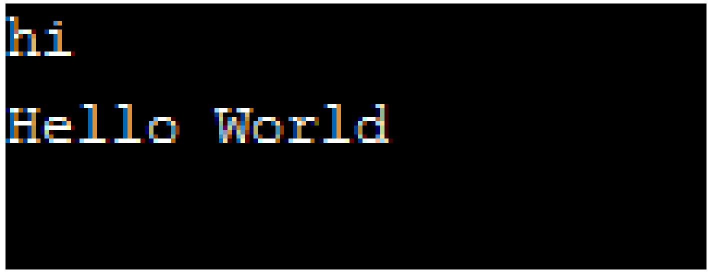
```
public class Main
{
    public static void main(String[] args) {

        int num=10;
        for (System.out.println("hi");num<=10;System.out.println("bye"))
        {

            System.out.println("Hello World");
            num=num+1;
        }}
}
```



```
hi
Hello World
bye
```

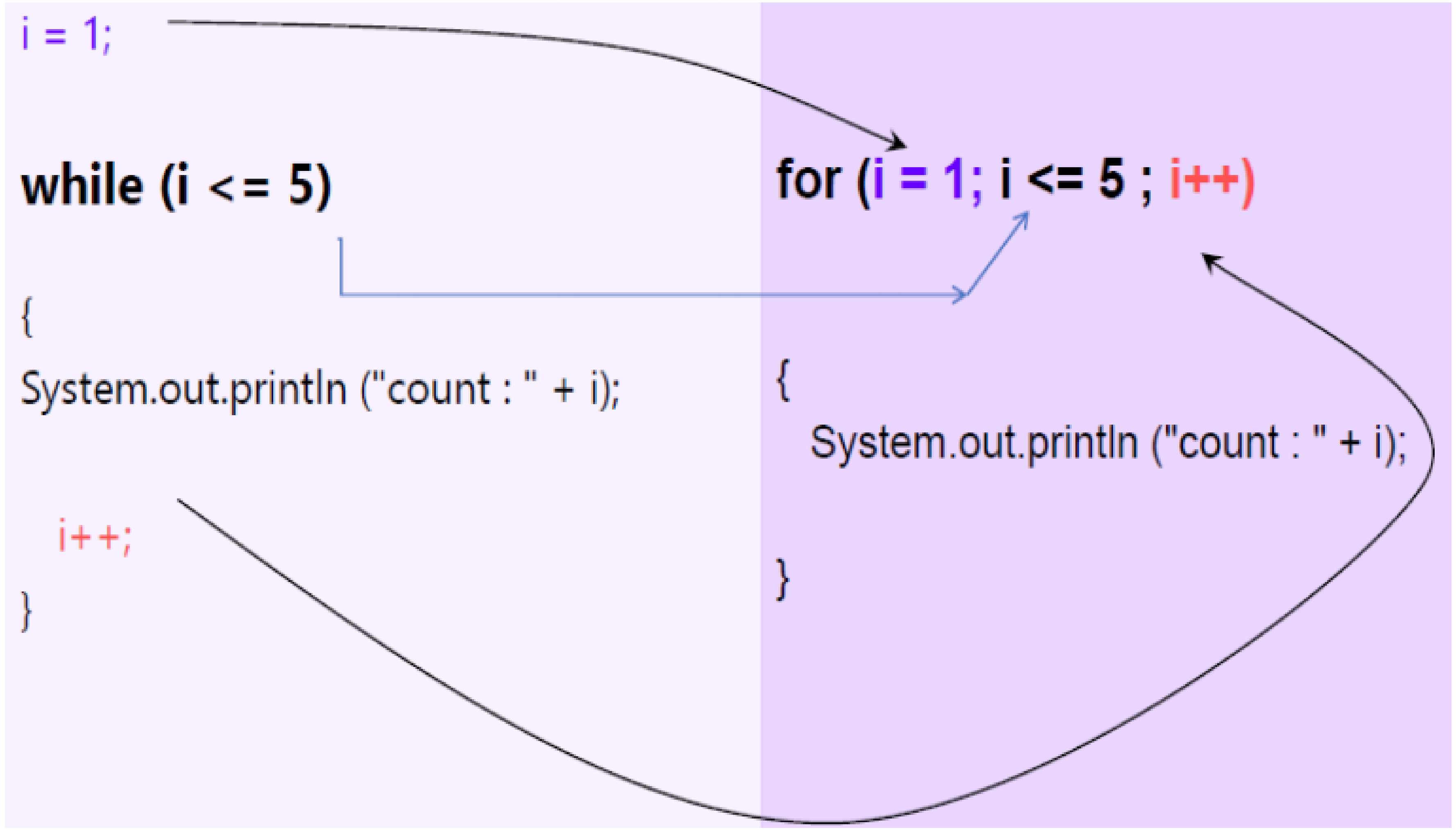


```
public class Main
{
    public static void main(String[] args) {

        int num=10;
        for (System.out.println("hi");true;System.out.println("bye"))
        {

            System.out.println("Hello World");
            break;
        }}
}
```



```
hi
Hello World
```

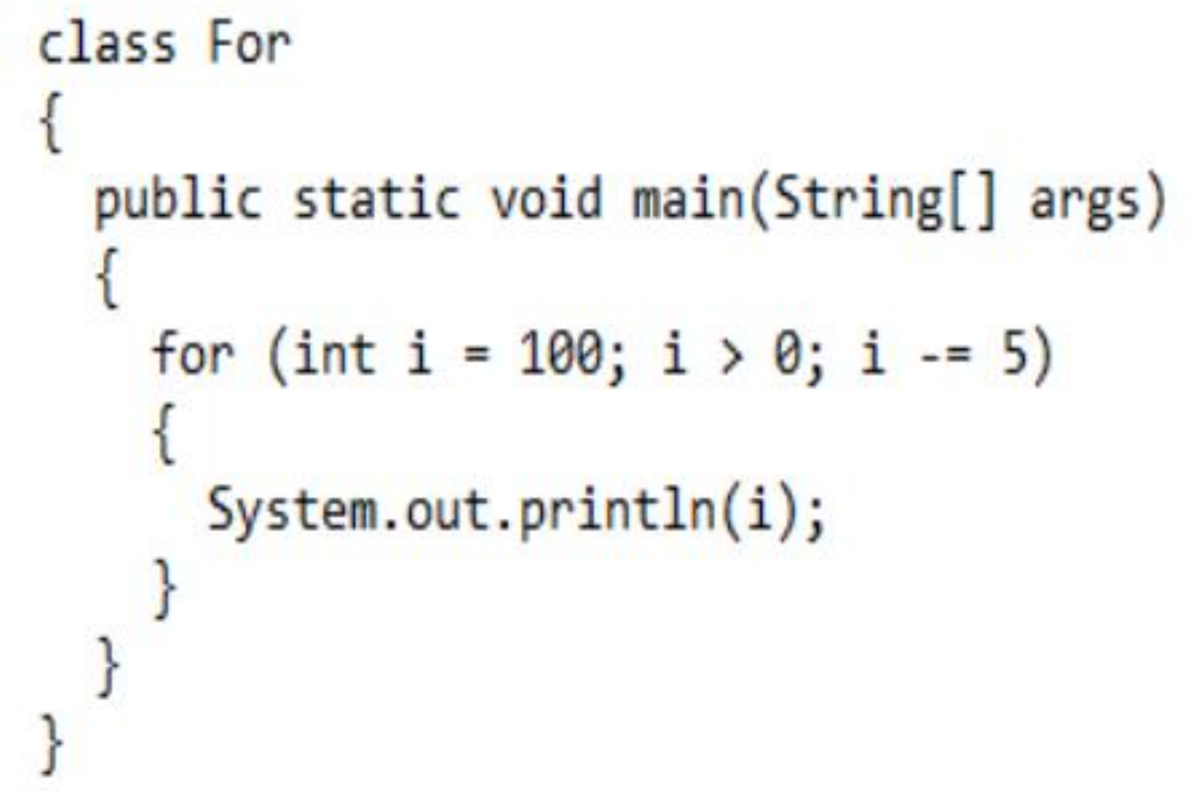
FOR LOOP (CONT..)



```
i = 1;

while (i <= 5)

{
System.out.println ("count : " + i);

   i++;
}
```

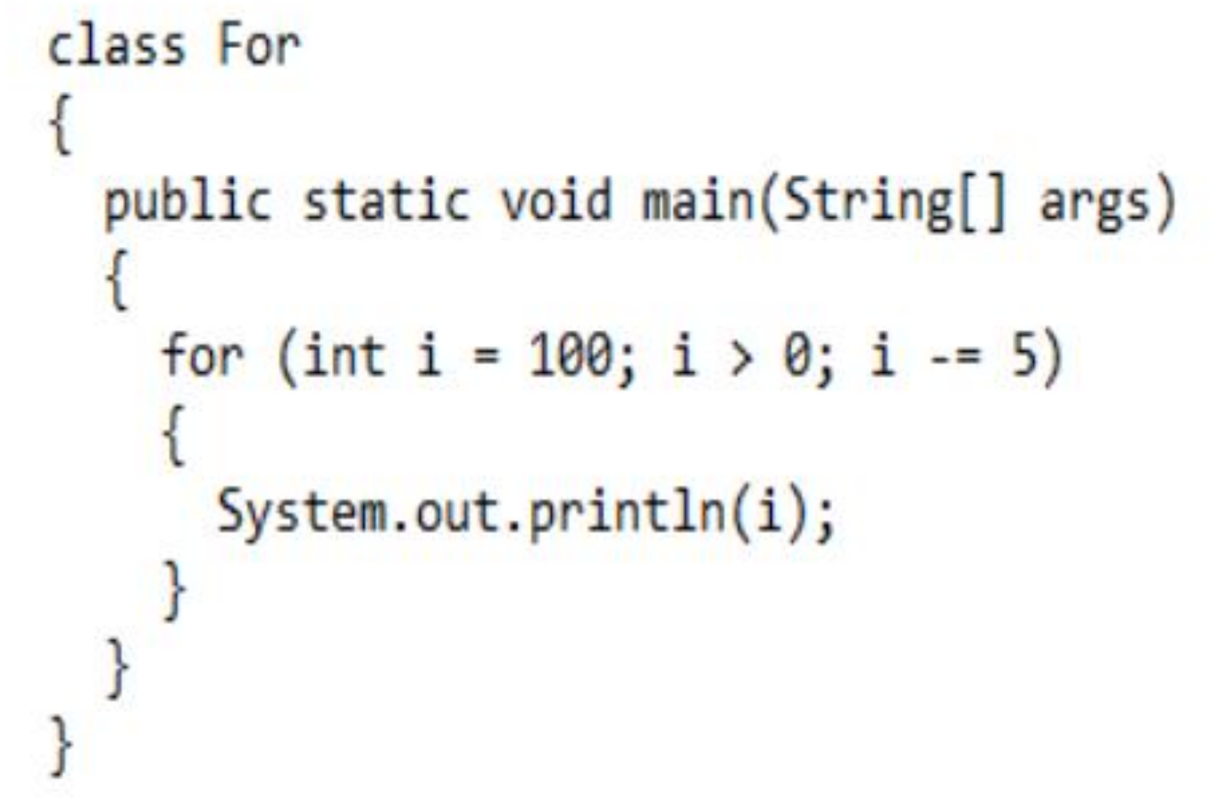
```
for (i = 1; i <= 5 ; i++)

{
   System.out.println ("count : " + i);

}
```

SIMPLE PROGRAM USING FOR LOOP



```
class For
{
  public static void main(String[] args)
  {
    for (int i = 100; i > 0; i -= 5)
    {
      System.out.println(i);
    }
  }
}
```

SIMPLE PROGRAM USING FOR LOOP



```
class For
{
  public static void main(String[] args)
  {
    for (int i = 100; i > 0; i -= 5)
    {
      System.out.println(i);
    }
  }
}
```

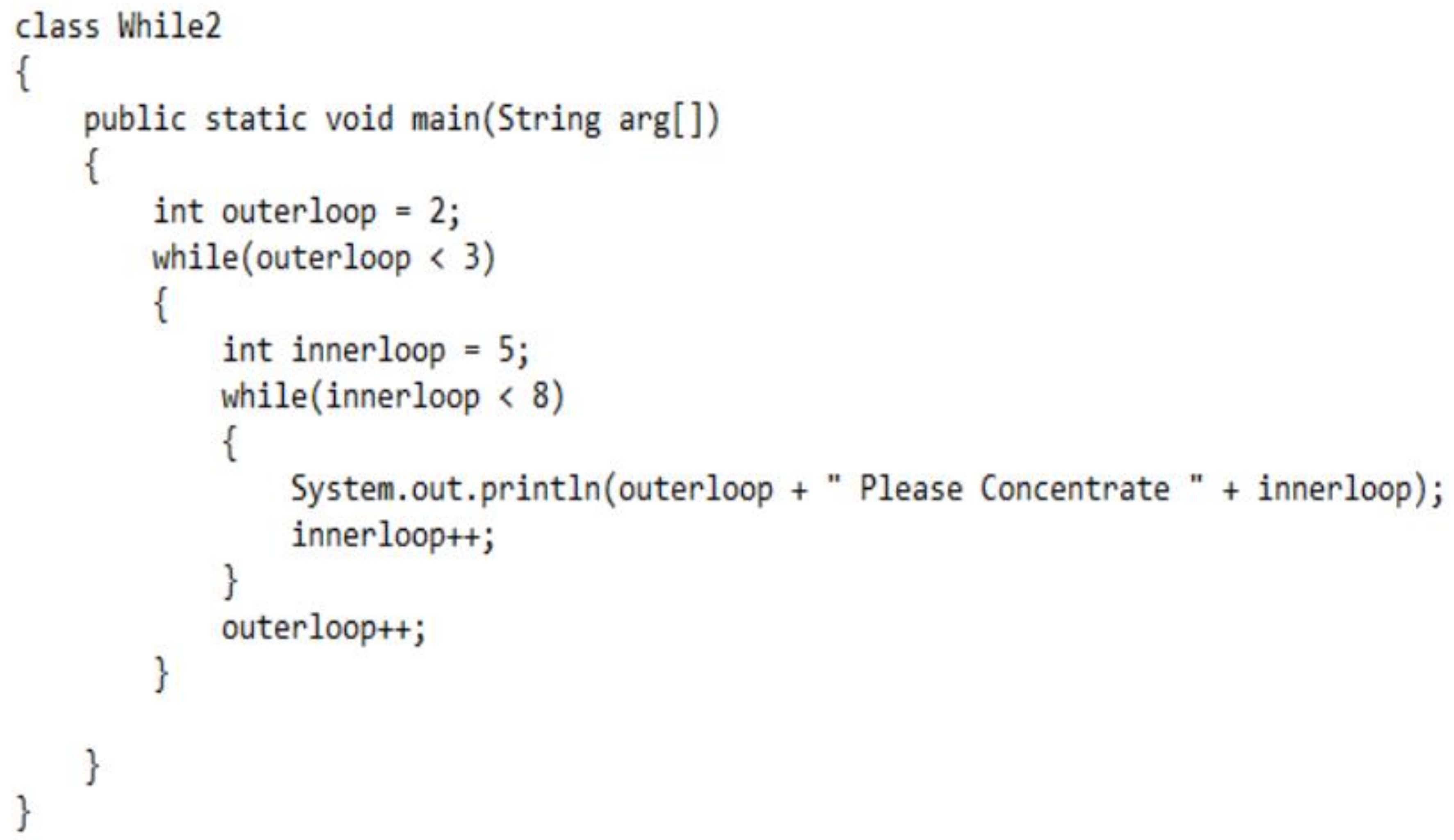
Output:

```
100
95
90
.
.
.
5
```

NESTED LOOP

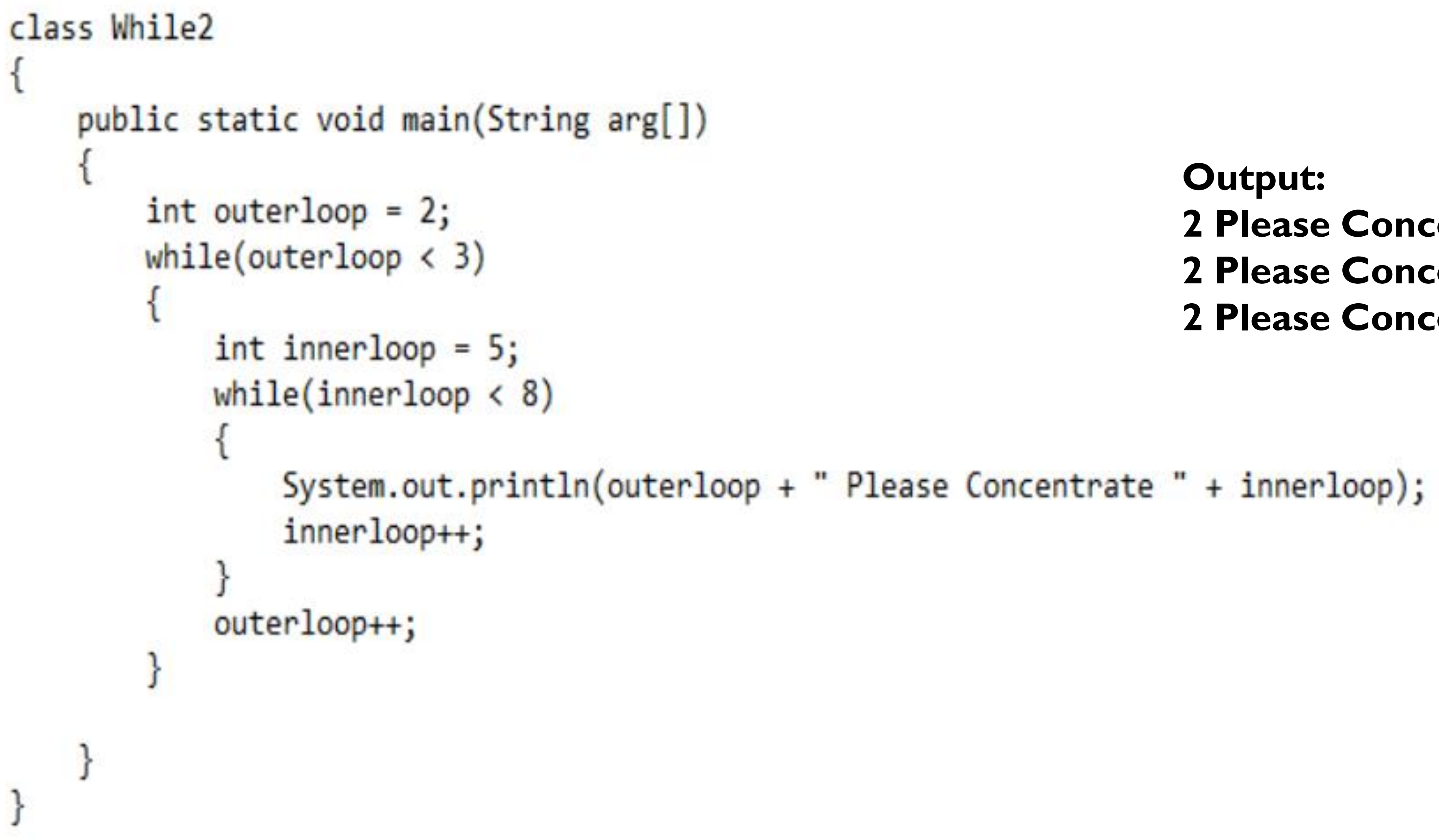
- Similar to nested if/else statements, loops can be nested as well.
- The body of a loop can contain another loop.
- For each iteration of the outer loop, the inner loop iterates completely .

SIMPLE PROGRAM USING NESTED LOOP



```
class While2
{
    public static void main(String arg[])
    {
        int outerloop = 2;
        while(outerloop < 3)
        {
            int innerloop = 5;
            while(innerloop < 8)
            {
                System.out.println(outerloop + " Please Concentrate " + innerloop);
                innerloop++;
            }
            outerloop++;
        }

    }
}
```

SIMPLE PROGRAM USING NESTED LOOP



```
class While2
{
    public static void main(String arg[])
    {
        int outerloop = 2;
        while(outerloop < 3)
        {
            int innerloop = 5;
            while(innerloop < 8)
            {
                System.out.println(outerloop + " Please Concentrate " + innerloop);
                innerloop++;
            }
            outerloop++;
        }

    }
}
```

Output:

```
2 Please Concentrate 5
2 Please Concentrate 6
2 Please Concentrate 7
```

TRANSFER STATEMENTS:

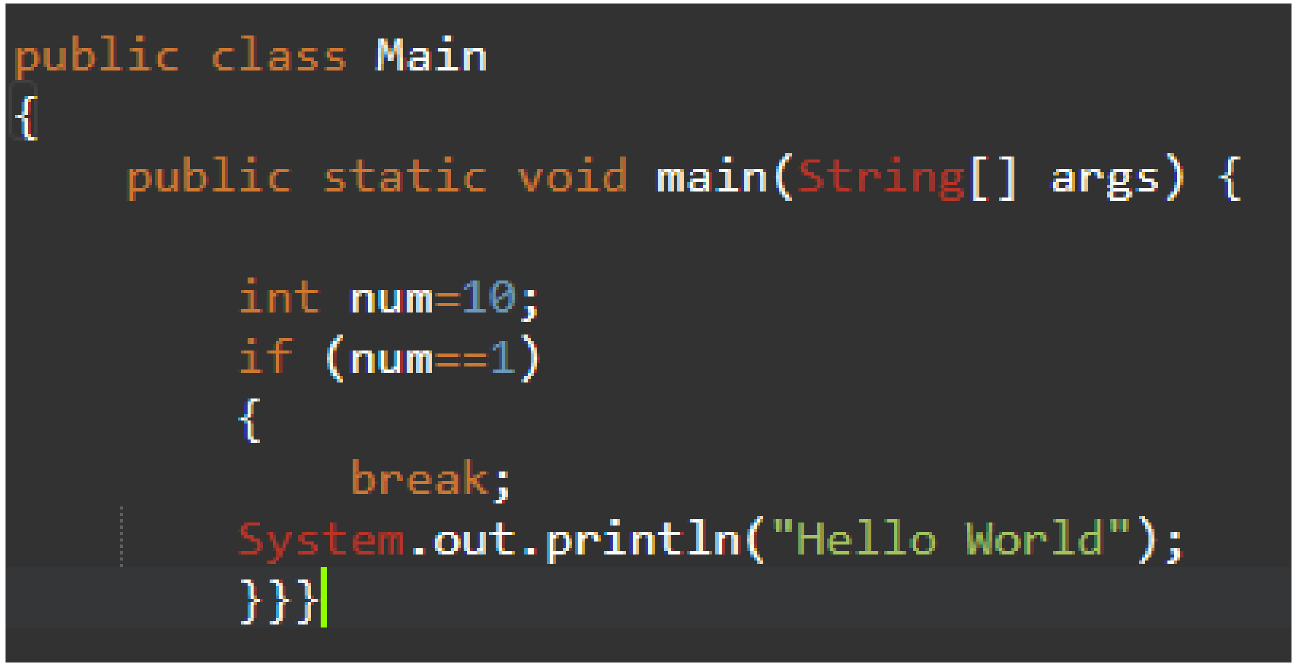
- **Break:**

We can use break statement for the following case.

- With in switch to stop fall through
- Inside loops to break the loop execution based on some condition.
- Inside labelled blocks to breaks that block execution based on some condition.
- Note: if we use break statement any where else we will get compile time error.

TRANSFER STATEMENTS:

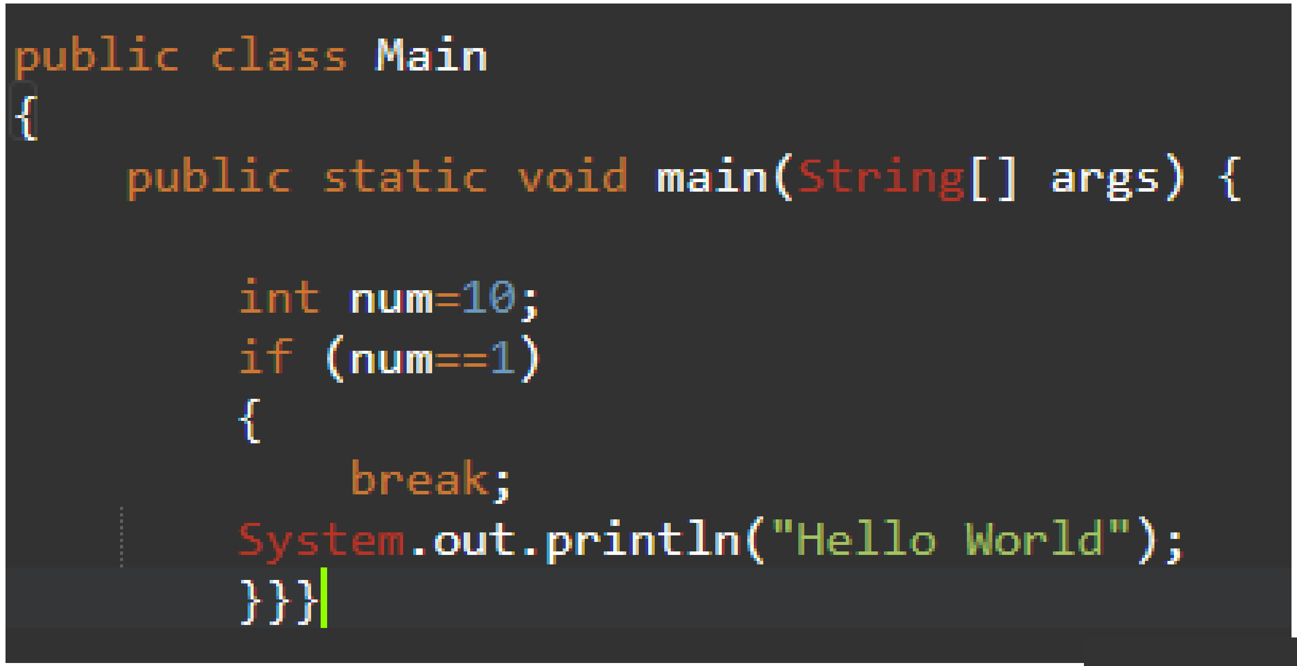
- **Break:**



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

        int num=10;
        if (num==1)
        {
            break;
        System.out.println("Hello World");
        }}}
```

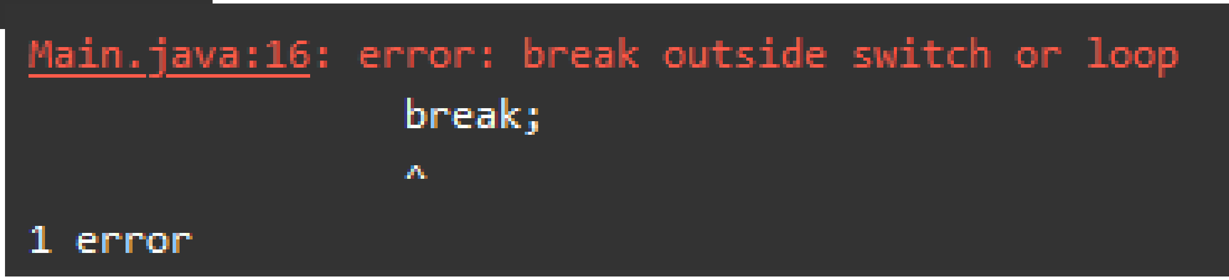
TRANSFER STATEMENTS:

- **Break:**



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

        int num=10;
        if (num==1)
        {
            break;
        System.out.println("Hello World");
        }}}
```

Output:



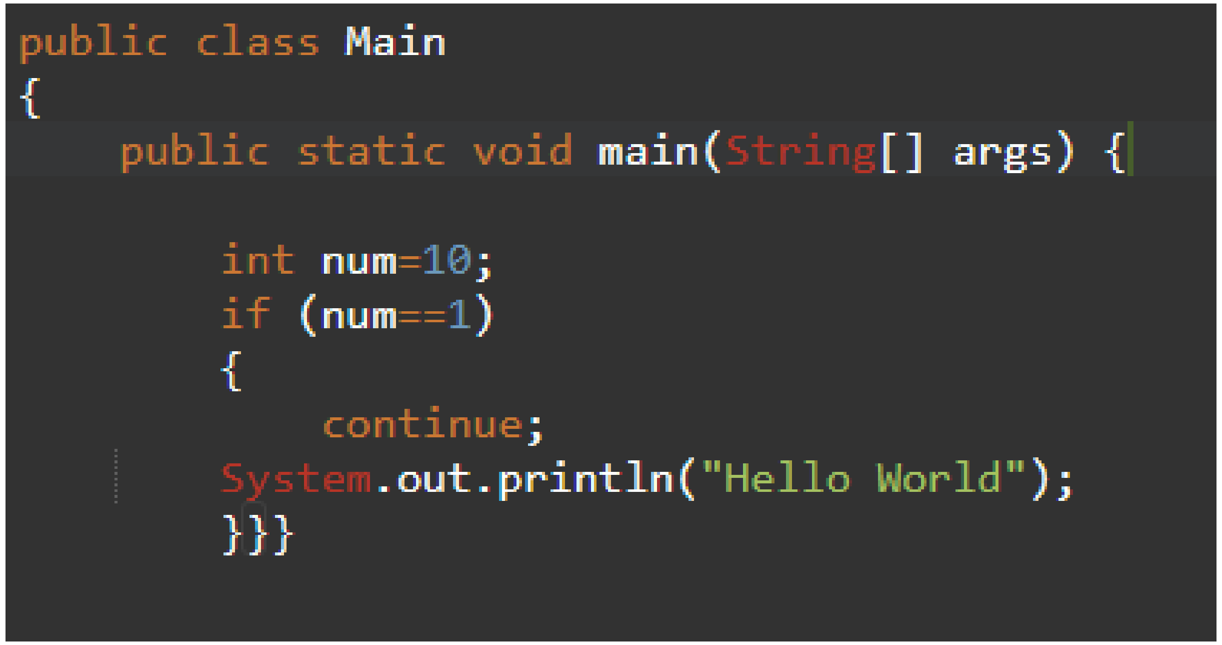
```
Main.java:16: error: break outside switch or loop
                break;
                ^
1 error
```

TRANSFER STATEMENTS:

- **Continue:**
  - **We can use continue statement to skip current iteration and continue for the next iteration inside loop.**
  - **If we are using continue outside of loops we will get compile time error.**

TRANSFER STATEMENTS:

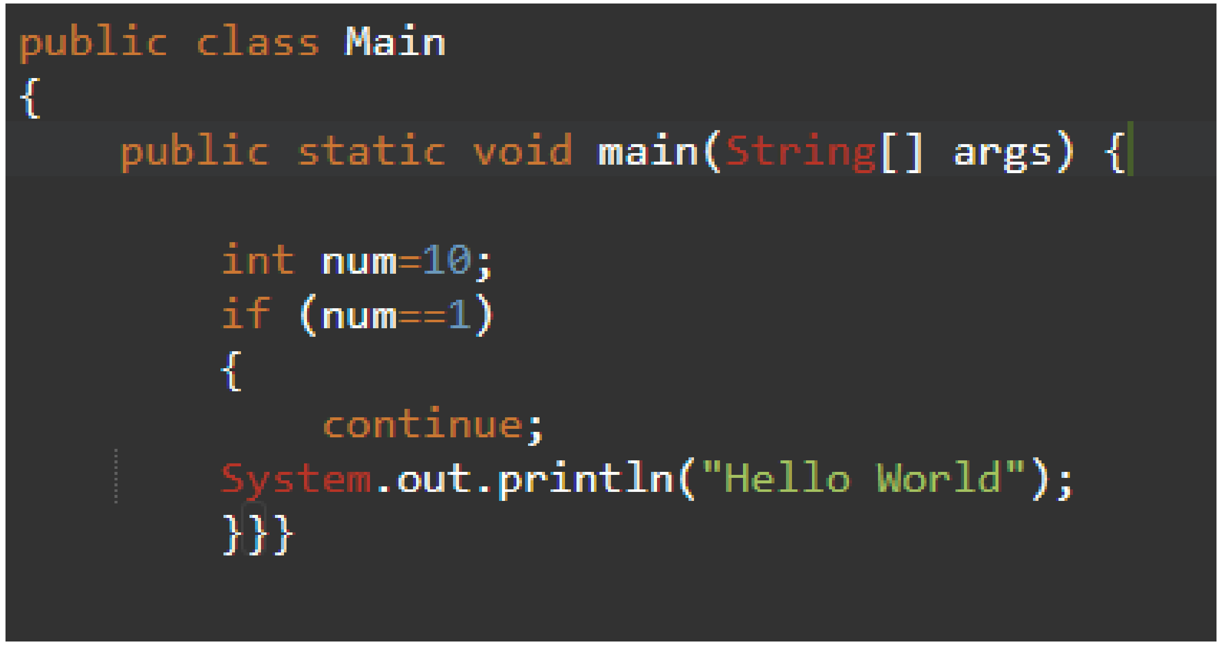
- **Continue:**



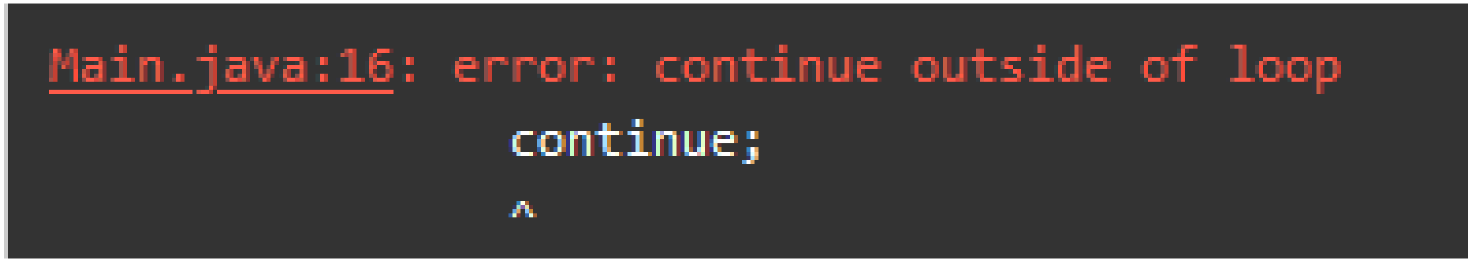
```
public class Main
{
    public static void main(String[] args) {

        int num=10;
        if (num==1)
        {
            continue;
        System.out.println("Hello World");
        }}}
```

TRANSFER STATEMENTS:

- **Continue:**



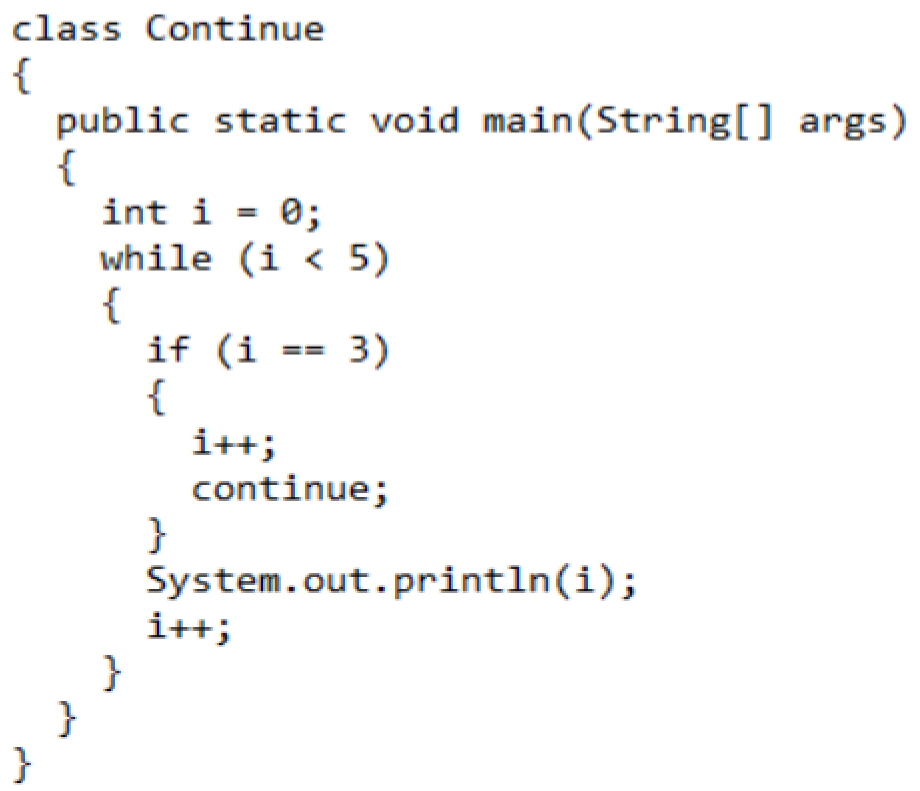
```
public class Main
{
    public static void main(String[] args) {

        int num=10;
        if (num==1)
        {
            continue;
        System.out.println("Hello World");
        }}}
```



```
Main.java:16: error: continue outside of loop
                    continue;
                    ^
```

TRANSFER STATEMENTS:

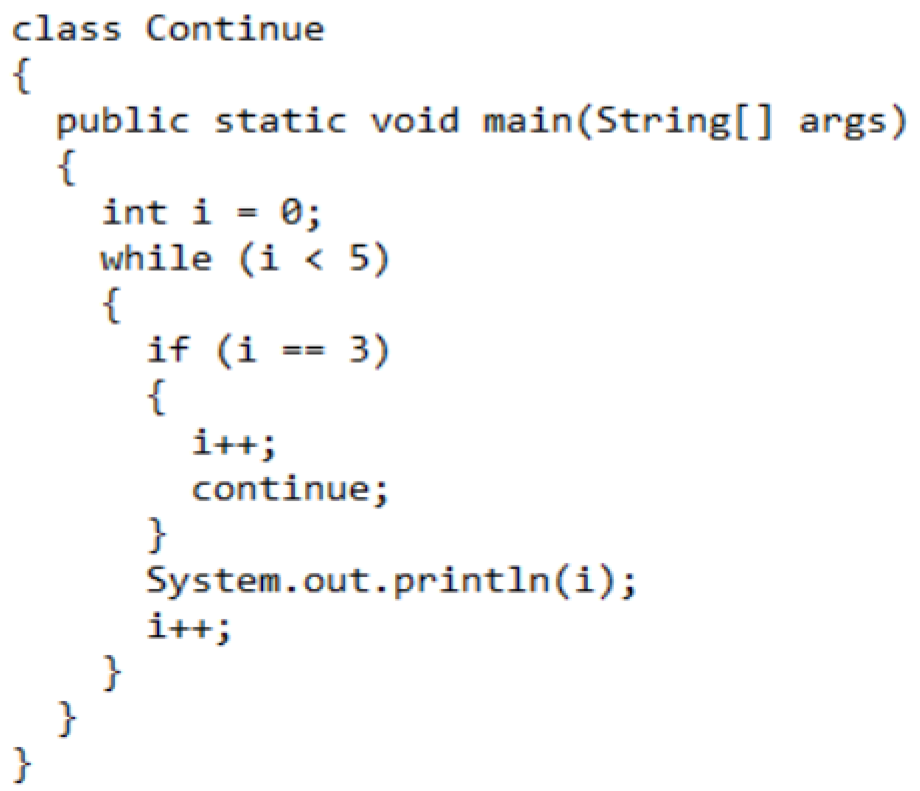
- **Continue:**



```
class Continue
{
  public static void main(String[] args)
  {
    int i = 0;
    while (i < 5)
    {
      if (i == 3)
      {
        i++;
        continue;
      }
      System.out.println(i);
      i++;
    }
  }
}
```

TRANSFER STATEMENTS:

- **Continue:**



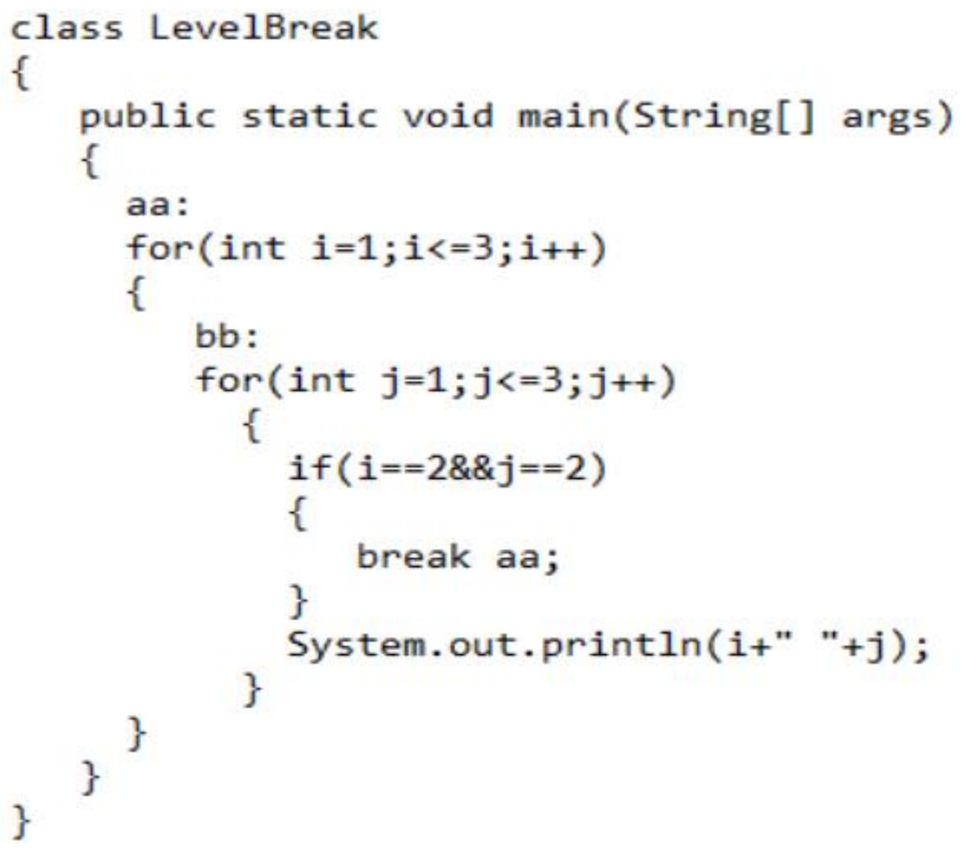
```
class Continue
{
  public static void main(String[] args)
  {
    int i = 0;
    while (i < 5)
    {
      if (i == 3)
      {
        i++;
        continue;
      }
      System.out.println(i);
      i++;
    }
  }
}
```

Output:

```
0
1
2
4
```

TRANSFER STATEMENTS:

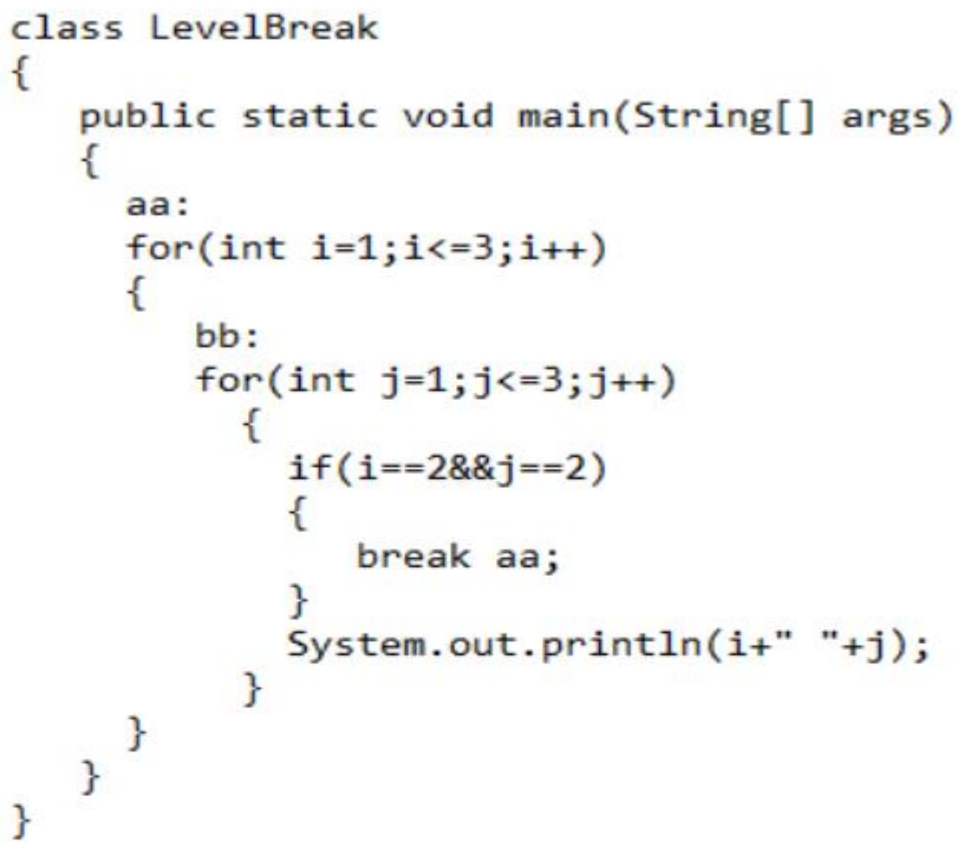
- **Label:**



```
class LevelBreak
{
    public static void main(String[] args)
    {
      aa:
      for(int i=1;i<=3;i++)
      {
          bb:
          for(int j=1;j<=3;j++)
            {
              if(i==2&&j==2)
              {
                  break aa;
              }
              System.out.println(i+" "+j);
            }
      }
    }
}
```

TRANSFER STATEMENTS:

- **Label:**



```
class LevelBreak
{
    public static void main(String[] args)
    {
      aa:
      for(int i=1;i<=3;i++)
      {
          bb:
          for(int j=1;j<=3;j++)
            {
              if(i==2&&j==2)
              {
                  break aa;
              }
              System.out.println(i+" "+j);
            }
      }
    }
}
```

Output:

```
1 1
1 2
1 3
2 1
```

THANK YOU

?