Flow Control

Switch Expression

```
// improved syntax for combining values
```

```
// Java 8
switch(a) {
  case 0:
  case 1:
  case 2:
    System.out.println("Good day");
    break;
  case 3:
  case 4:
    System.out.println("Hi");
    break;
  default:
    System.out.println("Hello");
    break;
}
```

```
// Java 17
switch(a) {
  case 0, 1, 2:
    System.out.println("Good day");
    break;
  case 3, 4:
    System.out.println("Hi");
    break;
  default:
    System.out.println("Hello");
    break;
}
```

"->" instead of ":"

no need for break statement

```
// preferred syntax in Java 17

switch(a) {
  case 0, 1, 2 -> System.out.println("Good day");
  case 3, 4 -> System.out.println("Hi");
  default -> System.out.println("Hello");
}

// multiple commands should be in the block code:
switch(a) {
  case 0, 1, 2 -> {
   isOK = true;
   System.out.println("Good day");
  }
  case 3, 4 -> System.out.println("Hi");
  default -> System.out.println("Hello");
}
```

```
// real improvement is that switch statement can be treated as an expression !!

String greeting = switch(a) {
                    case 0, 1, 2 -> "Good day";
                    case 3, 4 -> "Hi";
                    default -> "Hello";
                  };
System.out.println(greeting);
```

this expression returns String

```
String greeting = <expression>;
```

```java
// we can use yield keyword (similar to return statement in methods)


int a = 1;

String greeting = switch(a) {

                    case 0, 1, 2 -> {

                      String str1 = "Good";

                      String str2 = " day";

                      yield str1 + str2;

                    }

                    case 3, 4 -> "Hi";

                    default -> "Hello";

                  };

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

```
Good day
```

```
public void greet (int a, int b) {
  String greeting = switch (a) {
    case 0 -> "Good morning";
    case 1 -> {
      if (b > 0) yield "Good morning";
        else yield "Good afternoon";
    }
    case 2 -> "Good evening";
    default -> "Hello";
  };
  System.out.println(greeting);
}
greet(1, -1);
```

```
Good afternoon
```

```
// you can use yield in a single statement (not a good practice)


int a = 1;

String greeting = switch(a) {

                    case 0, 1, 2 -> "Good day";

                    case 3, 4 -> { yield "Hi"; }

                    default -> "Hello";

                  };

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

```
// switch expression can return different value types:


public void greet (int a) {

  var printOut = switch (a) {

                   case 0 -> "Good morning";   // String

                   case 1 -> 7;                // int

                   case 2 -> true;             // boolean

                   default -> 3.14;            // double

                 };

  System.out.println(printOut);

}
```

type will be determined at a runtime

```
// switch expression must handle all possible cases !!

public void greet (int a) {
  var printOut = switch (a) {
     case 0 -> "Good morning";
     case 1 -> "Good afternoon";
     case 2 -> "Good evening";
  };
  System.out.println(printOut);
}

// DOES NOT COMPILE
// fix: add default statement
// if we use enums, we can just list all possible values:

enum Compass {NORTH, SOUTH, EAST, WEST}

String getDirection (Compass value) {
  return switch(value) {
           case NORTH -> "Up";
           case SOUTH -> "Down";
           case EAST -> "Right";
           case WEST -> "Left";
         };
}
System.out.println(getDirection(Compass.SOUTH));
```

```
Down
```