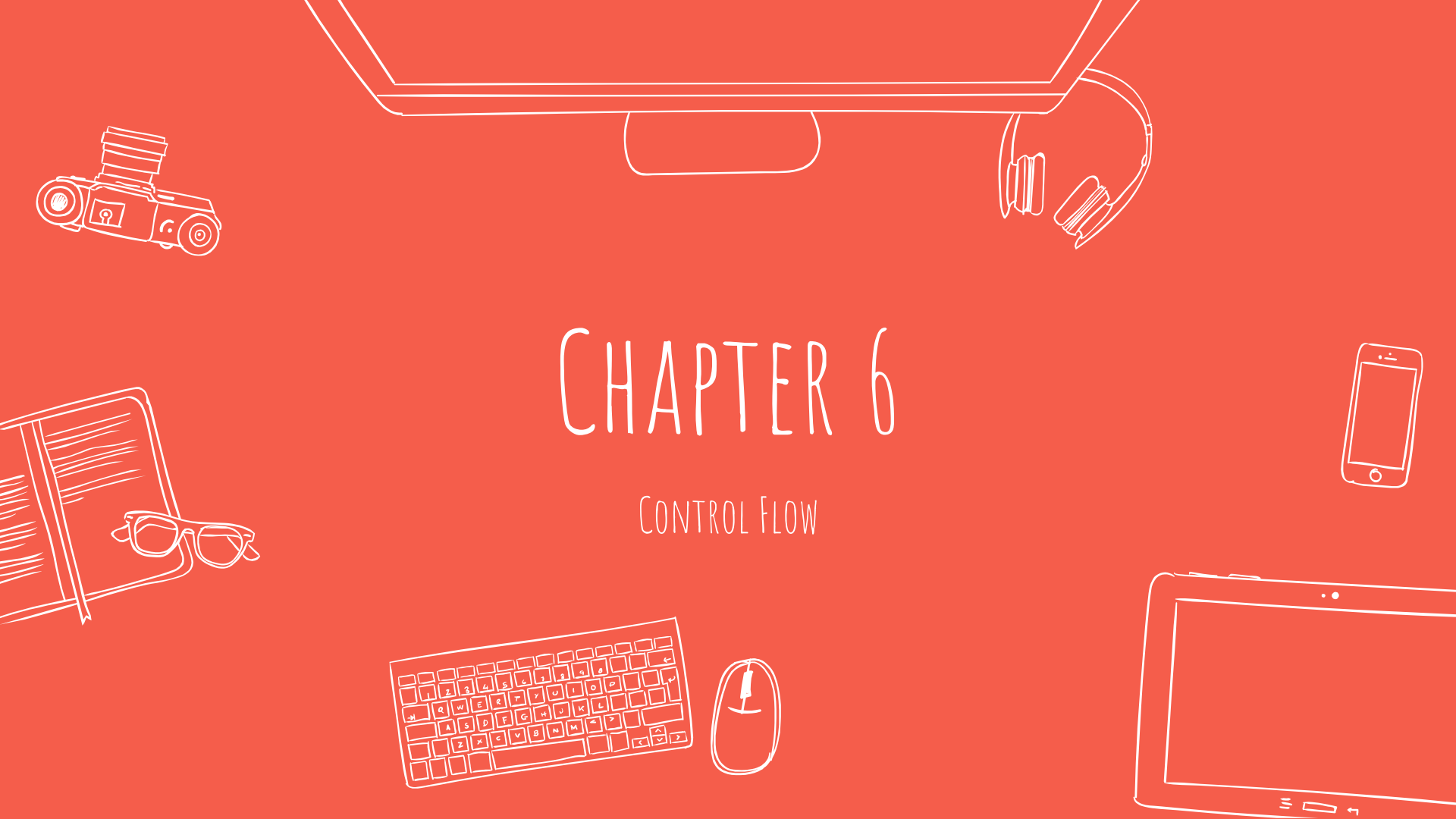
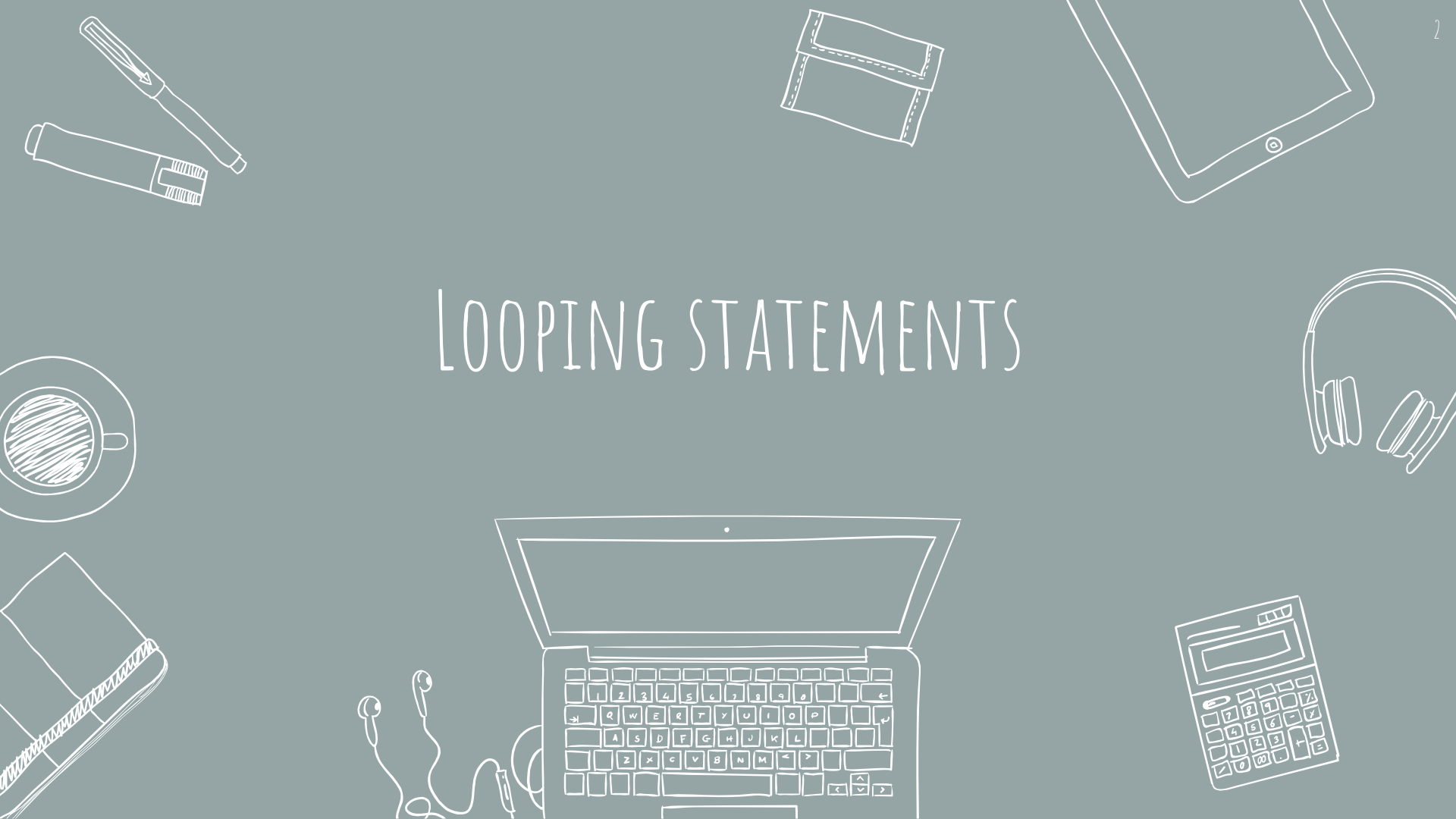


# Chapter 6

## Control Flow

# Looping statements



# Looping statements

While

do-while

For

# While

Syntax:

```
while (expression) {
    statement(s)
}
```

E.g.

```
while (userOption == 'E') {
        System.out.println ("Perform operation " + userOption);
}
```

# DO-WHILE

Syntax:

```
do {
    statement(s)
} while (expression);
```

Do block is executed at least once.

E.g.

```
int i=10;
do {
        System.out.println(i);
        i--;
} while(i>1);
```

# For Loop

Syntax:

```
for (initialization; termination; increment) {
    statement(s)
}
```

E.g.

```
for (int i=0; i < 10; i++) {
            System.out.println(”Value “ + i);
}
```

# For Each

Syntax:

```
for (collectionElement : collection ) {
    statement(s)
}
```

E.g.

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (String car : cars) {
        System.out.println(”Car name “ + car);
}
```

# Branching statements

Break

Continue

Return

Labeled break

# Break statement

Break statement terminates the innermost switch, for, while, or do-while statement

E.g.

```
while (true) {
        userInput = readUserInput();
        if(userInput == 'E') {
            // This will exit while loop and continue with next statement after while block.
            break;
        }
        System.out.println("Continue processing");
}
```

# Continue statement

Continue statement is used inside loops. Whenever it is encountered, control directly jumps to the beginning of the loop for **next** iteration, skipping the execution of statements inside loop's body for the current iteration

E.g.

```
for (int i =0; i < 10; i++) {
  if (isPrimeNumber(i)) {
    continue;
  }
  printFactors(i);
}
```

# Return statement

A return statement causes the program control to transfer back to the caller of a method. Every method in Java is declared with a return type and it is mandatory for all java methods. Return statement in loops cause to terminate the loop and control get transfer to caller of the method.

```
public static boolean studentPassTheExam (int[]  aStudentMarks) {
  for (int mark : aStudentMarks) {
    if (mark < 50) {
      return false;
    }
  }
return true;
}

public static void main () {
  int[] studentMarks = {67, 90, 54, 25};
  if(studentPassTheExam (studentMarks)) {
    System.out.println("Pass");
  }
}
```

# Labeled break statement

labeled break terminates an outer statement.  it does not transfer the flow of control to the label. Control flow is transferred to the statement immediately following the labeled (terminated) statement.

# Quiz

What output do you think the code will produce if aNumber is 3?

```
if (aNumber >= 0)
    if (aNumber == 0)
        System.out.println("first string");
else System.out.println("second string");
System.out.println("third string");
```

# Thanks!

## Any questions?

You can find me at:

vijay_garry@hotmail.com

https://github.com/vijaygarry