
ORACLE
ACADEMY

Java Foundations

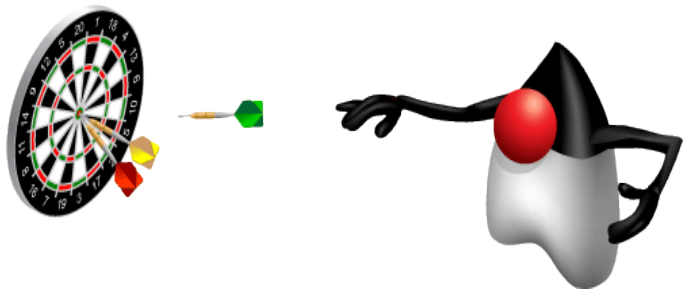
6-3
Using `break` and `continue` Statements

ORACLE ACADEMY

Objectives

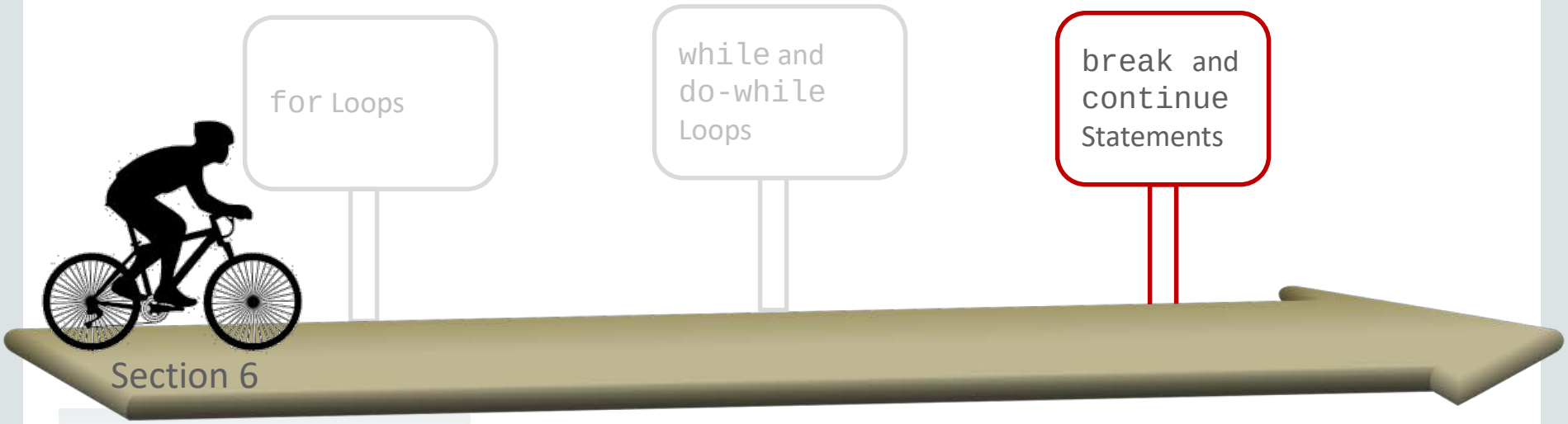
This lesson covers the following objectives:

- Use a `break` statement to exit a loop
- Use a `continue` statement to skip part of a loop
- Explain the need for loop comments

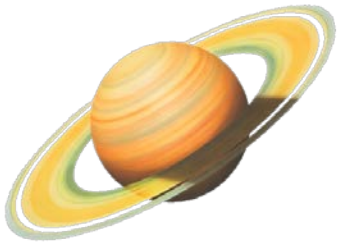


Topics

- Using a `break` Statement in a Loop
- Using a `continue` Statement in a Loop
- Writing Loop Comments



Mission to Saturn's Rings



- Let's consider another scenario for this mission. As the spaceship is rotating around Saturn and taking snapshots, the robotic arm or camera breaks.
- How would you solve this problem?
  - If you were to write a Java program, which construct would you use?
  - Let's see whether Java has a statement that enables to you to end a loop immediately

How Do You Exit a Loop Early?

- Usually, the only way to exit a loop is for the loop condition to evaluate to false.
- However, it's often convenient to terminate a loop early when certain conditions are met.
- In such cases, continuing to loop would be a waste of processor time.

How Do You Exit a Loop Early?

You can use two Java statements to terminate a loop early:

- `break`
- `continue`

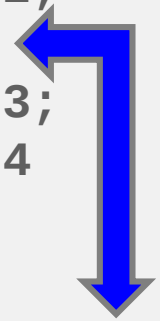
Using `break` in a Loop

- When a `break` statement is executed inside a loop, the loop statement is terminated immediately.
- The program continues to execute with the statement following the loop statement.
- Syntax:

```
break;
```

Using `break` in a `while` Loop

```
while(condition){
    statement1;
    statement2;
    break;
    statement3;
    statement4
}
statement;
```



Control passes to the statement outside the loop.

[statement outside the `while` loop]

Using `break` in a `while` Loop: Example

```
public static void main(String[] args) {
    int i = 0;
    while (i < 10) {
        System.out.println(i + "\t");
        i++;
        if (i == 4) {
            break;
        }
    }
}
```

Output: 0 1 2 3

Execution of the loop is terminated when the loop counter is equal to 4.

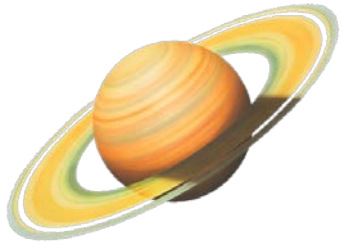
Using `break` in a `for` Loop

- Let's write a program to demonstrate a `break` statement in a `for` loop.
- The program must …
  - Read 10 numbers from the console.
  - Compute the sum of the numbers that the user enters.
  - If the user enters 999, terminate the loop regardless of the value of the loop counter and without adding to the sum.

Using break in a for Loop: Example

```
public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    int numInputs = 10, input = 0, sum = 0, stopLoop = 999;
    System.out.println(“Enter 10 numbers”);
    for (int i = 0; i < numInputs; i++) {
        input = in.nextInt();
        if (input == stopLoop){
            break;
        }
        else {
            sum += input;
        }
    }
    System.out.println("The sum of the numbers:" + sum);
}
```

Mission to Saturn's Rings: Implementing the Conditions



Let's use a `while` loop and a `break` statement to implement the conditions specified at the beginning of the lesson.

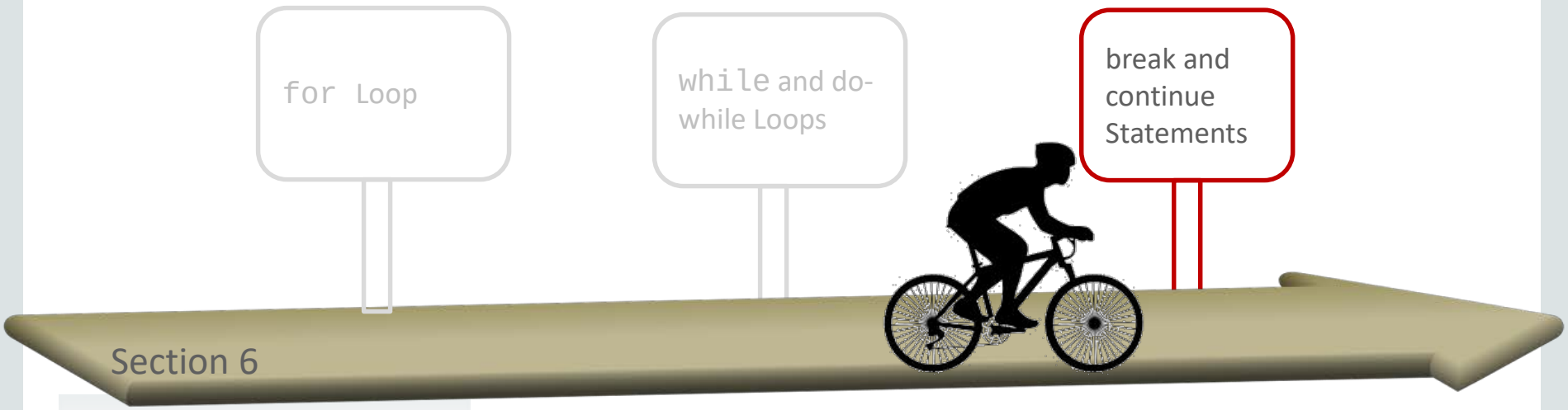
```
public static void main(String[] args) {
    long distTravelled = 0;
    long minDistance = 50000000;
    while (distTravelled >= minDistance) {
        snap++; //click snap
        if (camera == broken) {
            break;
        }
        else {
            rotate();
        }
    }
}
```

Exercise 1

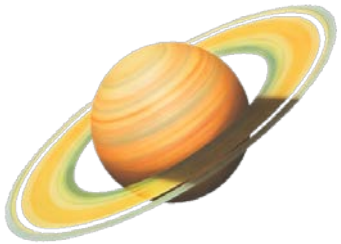
- Import and open the `BreakContinueEx` project.
- Examine `ComputeSum.java`.
- Implement the following:
  - Accept 10 numbers from the user.
  - Compute the sum of the numbers entered.
  - When 0 is entered, the program must exit and display the sum of the numbers.

Topics

- Using a `break` Statement in a Loop
- Using a `continue` Statement in a Loop
- Writing Loop Comments



Mission to Saturn's Rings: Another Scenario



- Let's consider another scenario for this mission. As the spaceship is rotating around Saturn and taking snapshots of Saturn's rings …
  - If the visibility is zero, do **not** take snapshots.
  - Otherwise, continue to take the snapshots.
- How would you solve this problem?
  - If you were to write a Java program, which construct would you use?
  - Let's see whether Java has a statement that enables you to skip the current iteration of the loop.

Using `continue` in a Loop

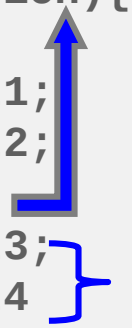
- Sometimes, you may want to skip the current iteration in a loop and not terminate the loop itself.
- You can use a `continue` statement to skip the current iteration in a loop:
  - That is, the rest of the loop body is skipped to the end of the loop. However, it doesn't end the loop.
  - When the program reaches the end of the loop, the program jumps back to test the loop continuation condition.
- Syntax:

```
continue;
```

Using `continue` in a `while` Loop

```
while(condition){

   statement1;
   statement2;
   continue;
   statement3;
   statement4
}
 statement;
```



Control passes to the loop condition.

These statements are skipped in the current iteration.

statement outside the `while` loop

Using `continue` in a `for` Loop

```
for (i = 0; i < 10; i++) {
    statement1;
    statement2;
    continue;
    statement3;
    statement4;
}
```

Control passes to the loop condition.

These statements are skipped in the current iteration.

Using `continue` in a `for` Loop

```
public static void main(String[] args) {
    for (int i = 0; i < 10; i++) {
        if (i == 4) {
            continue; //control jumps to update i++
        }
        System.out.print(i + "\t");
   }
}
```

Output: 0 1 2 3 5 6 7 8 9

The output doesn't include 4. Because of the `continue` statement, the loop execution is skipped when the loop counter is 4.

Putting It All Together

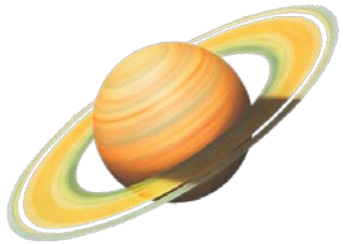
- Let's write a program using the `while` loop and the `continue` statement.
- The program must …
  - Compute the sum of numbers between 1 and 99 using the `while` loop.
  - If the number is a multiple of 10, the current iteration must be skipped and the number must not be added to the sum.
  - Display the sum to the console.

Computing the Sum of Numbers

```
public static void main(String[] args) {
    int counter = 0;
    int sum = 0;
    while (counter < 100) {
        counter++;
        if (counter % 10 == 0) {
            continue;
        }
        else {
            sum += counter;
        }
    }
    System.out.println("Sum of 1 - 99: " + sum);
}
```

Is this a multiple of 10? If yes, then skip the current iteration.

Mission to Saturn's Rings: Implementing the Conditions



Let's use a `while` loop and a `continue` statement to implement the conditions specified at the beginning of this topic.

```
public static void main(String[] args) {
    long distTravelled = 0;
    long minDistance=50000000;
    while (distTravelled >= minDistance) {
        if (visibility == 0) {
            continue;
        }
        else {
            snap++;
        }
   }
}
```

Exercise 2

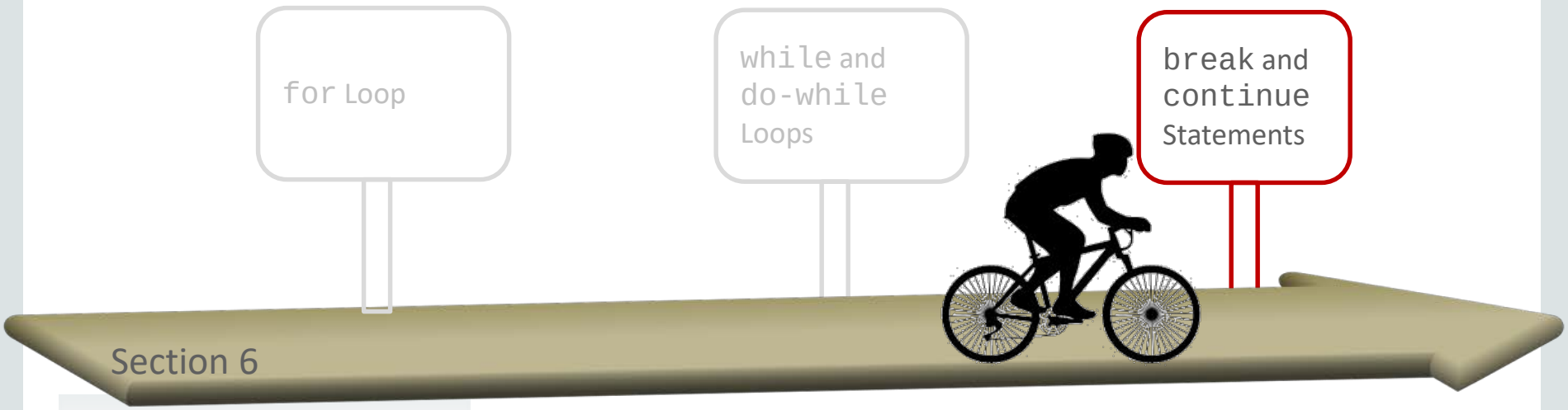
- Import and open the `BreakContinueEx` project.
- Examine `CountChar.java`.
  - The program is used to count the number of occurrences of the char ‘w’ in the string.
  - Modify the program to …
    - Resolve the syntax error
    - Print the count of char ‘w’
  - Expected Output:
    - Number of w : 3

Exercise 3

- Import and open the `BreakContinueEx` project.
- Examine `BreakContinue.java`.
- Modify the program by using `break` and `continue` statements …
  - If the number is even, the number should not be printed.
  - Execution of the loop should stop when the value of the loop counter is 7.

Topics

- Using a `break` Statement in a Loop
- Using a `continue` Statement in a Loop
- Writing Loop Comments



Section 6

Writing Loop Comments

- It's a good practice to add appropriate comments to loops. Otherwise …
  - Code tends to be confusing to look at.
  - You won't be able to understand the logic very easily.
- It helps to understand …
  - Loop variables used and their purpose
  - Logic of the loop
  - Number of iterations
  - Execution of the statements in the loop depending on the condition or criteria or both

Writing Loop Comments: Example

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

    Scanner in = new Scanner(System.in);
    int numInputs = 10, input = 0;

    // This loop is executed 10 times
    for (int i = 0; i < numInputs; i++) {
        input = in.nextInt();    //user inputs a number


    if (input % 2 == 0) {        //if the number is even skip the
        continue;               //remaining code and restart the loop
    }

    System.out.println("That number was odd");
    }
}
```

Exercise 4

- Import and open the `BreakContinueEx` project.
- Examine `Divisors.java`.
- The program finds all divisors of a number.

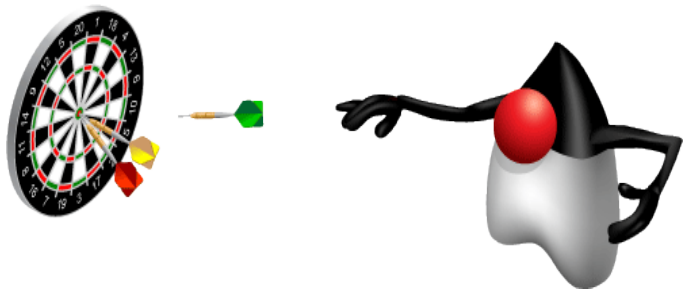
Exercise 4

- Modify the program to include comments for the loop about …
  - Loop variables used
  - Logic of the loop
  - Number of iterations
  - Condition used
  - Control flow in the loop

Summary

In this lesson, you should have learned how to:

- Use a `break` statement to exit a loop
- Use a `continue` statement to skip part of a loop
- Explain the need for loop comments




ORACLE
ACADEMY