Unit 4: Iteration
For Loops

Adapted from:

1) Building Java Programs: A Back to Basics Approach

by Stuart Reges and Marty Stepp

2) Runestone CSAwesome Curriculum

Textbook Reference

Online Textbook Think Java - 2nd Edition by Allen Downey and Chris Mayfield

For this lecture use Chapter 6.3

Categories of loops

indefinite loop: One where the number of times its body repeats is not known in advance.

- Prompt the user until they type a non-negative number.
- Print random numbers until a prime number is printed.
- Repeat until the user has types "q" to quit.

The **while loop** is usually used for indefinite loops.

Categories of loops

definite loop: Executes a known number of times.

- Print "hello" 10 times.
- Find all the prime numbers up to an integer n.
- Print each odd number between 5 and 127.

In this lecture, we will see that a for loop is often used to implement a definite loop.

Repetition with `for` loops
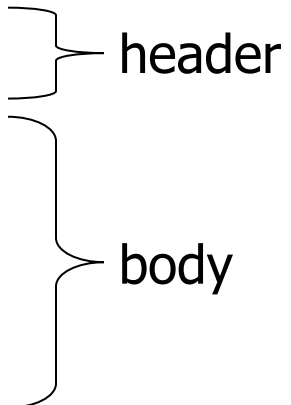
- So far, repeating a statement is redundant:

```
System.out.println("Homer says:");
System.out.println("I am so smart");
System.out.println("I am so smart");
System.out.println("I am so smart");
System.out.println("I am so smart");
System.out.println("S-M-R-T... I mean S-M-A-R-T");
```

- Java's `for` **loop** statement performs a task many times.

```
System.out.println("Homer says:");
for (int i = 1; i <= 4; i++) {    // repeat 4 times
    System.out.println("I am so smart");
}
System.out.println("S-M-R-T... I mean S-M-A-R-T");
```

for loop syntax

```
for (initialization; test; update) {      header
    statement;
    statement;

    ...
    statement;                            body
}
```

– Perform **initialization** once.
– Repeat the following:
  - Check if the **test** is true. If not, stop.
  - Execute the **statement**s.
  - Perform the **update**.

Initialization

```
for (int i = 1; i <= 6; i++) {
    System.out.println("I am so smart");
}
```

- Tells Java what variable to use in the loop
  - Performed once as the loop begins

  - The variable is called a loop counter
    - can use any name, not just `i`
    - can start at any value, not just `1`

Test

```
for (int i = 1; i <= 6; i++) {
    System.out.println("I am so smart");
}
```

- Tests the loop counter variable against a limit
  - Uses comparison operators:
    - $<$ less than
    - $<=$ less than or equal to
    - $>$ greater than
    - $>=$ greater than or equal to

Repetition over a range

Use a for loop to print:

```
System.out.println("1 squared = " + 1 * 1);
System.out.println("2 squared = " + 2 * 2);
System.out.println("3 squared = " + 3 * 3);
System.out.println("4 squared = " + 4 * 4);
System.out.println("5 squared = " + 5 * 5);
System.out.println("6 squared = " + 6 * 6);
```

– Intuition: "I want to print a line for each number from 1 to 6"

Answer

```
for (int i = 1; i <= 6; i++) {
    System.out.println(i + " squared = " + i * i);
}
System.out.println("Whoo!");
```

Output:

```
1 squared = 1
2 squared = 4
3 squared = 9
4 squared = 16
5 squared = 25
6 squared = 36

Whoo!
```

Example

Write a for loop to print: 5 9 13 17 21 25

Answer:

```
for (int i = 5; i <= 25; i += 4)
        System.out.print(i + " ");
    }
```

Counting down

- The **update** can use `--` to make the loop count down.
  - The **test** must say `>` instead of `<`

```
System.out.print("T-minus ");
for (int i = 10; i >= 1; i--) {
    System.out.print(i + ", ");
}
System.out.println("blastoff!");
System.out.println("The end.");
```

  - Output:

```
T-minus 10, 9, 8, 7, 6, 5, 4, 3, 2, 1,
blastoff!
The end.
```

For Loop in Movies

Groundhog Day(1993); Bill Murray.
Looper(2010); Bruce Willis and Joseph
 Gordon-Levitt, Emily Blunt.
Edge of Tomorrow(2014); Tom Cruise,
 Emily Blunt.
Happy Death Day(2017).

TV-Shows:
Russian Doll(Netflix, Emmy-nominated)

isPrime

```java
public static boolean isPrime(int n) {
    int factors = 0;
    for (int i = 1; i <= n; i++) {
        if (n % i == 0) {
            factors++;
        }
    }
    if (factors == 2) {
        return true;
    } else {
        return false;
    }
}
```

- Calls to methods returning `boolean` can be used as tests:

```java
if (isPrime(57)) {
    ...
}
```

The condition below is verbose. Can we simplify it? Replace it with one statement!

```java
public static boolean isPrime(int n) {
    int factors = 0;
    for (int i = 1; i <= n; i++) {
        if (n % i == 0) {
            factors++;
        }
    }

        if(factors == 2)
            return true;
        else
            return false;

}
```

Improved `isPrime` method

The following version utilizes Boolean Zen:

```
public static boolean isPrime(int n) {
    int factors = 0;
    for (int i = 1; i <= n; i++) {
        if (n % i == 0) {
            factors++;
        }
    }
    return factors == 2;  // if n has 2 factors, true
}
```

For vs While

Write a loop to compute the sum:

$$1 + 2 + 3 + \ldots + 99 + 100$$

```
int sum = 0;
int number = 1;
while(number <= 100){
    sum += number;
    number++;
}
```

Both for and while loops can be used to solve this problem.

```
int sum = 0;
for(int i = 1; i <= 100; i++){
    sum += i;
}
```

For vs. While

Although for and while loop can generally be interchangeable. It is best practice to use a for loop as a definite loop where the beginning and termination of the loop is well defined.

```
for(int i = 1; i <= 100; i++){
      System.out.println(i);
}
```

This for loop executes 100 times. It is a definite loop. It is usually better to use a for loop as a definite loop.

For vs. While

If the termination condition of a loop is less well defined, use a while loop. For example, suppose you require a positive integer input from the user. The following will loop until a positive number is entered.

```
Scanner inp = new Scanner(System.in);
System.out.print("Enter a positive number:");
int x = inp.nextInt();


while(x <= 0){
        System.out.print("Enter a positive number:");
        x = inp.nextInt();
}
System.out.print("Thank you!");
```

Sample Output:
Enter a positive number: -4
Enter a positive number: -5
Enter a positive number: -10
Enter a positive number: 6
Thank you!

This while loop executes an unknown number of times since you don't know when the user will enter a positive number. It is a indefinite loop. It is better to use a while loop here.

String Algorithms: For Loops

Loops allow us to do String traversals. Suppose we want to know the number of spaces in a String.

```
public static int countSpaces(String str){
        int count = 0;
        for(int i = 0; i < str.length(); i++){
                if(str.substring(i, i + 1).equals(" "))
                        count++;
        }
        return count;
}
```

String Algorithms: For Loops

Assume that a given String is a sequence of words separated by spaces, return the number of words. Use countSpaces.

```
public static int numOfWords(String str){
        return countSpaces(str) + 1;
}
```

Remove Spaces (While)

Given a string, return the string with all of its spaces removed.

We will do this first with a **while loop**.
One way to do this is to use a while loop with indexOf.

```java
public static String removeSpaces1(String str){
    while(str.indexOf(" ") != -1){
        int indexSpace = str.indexOf(" ");
        String first = str.substring(0, indexSpace);
        String second = str.substring(indexSpace + 1);
        str = first + second; // first space removed
    }
    return str;
}
```

Remove Spaces (For)

Given a string, return the string with all of its spaces removed.

We now implement this with a **for loop**. Start with an empty string. Then examine every letter and if it is NOT a space, add it to the string. This builds up the String one letter at a time but avoids the spaces. For some students, this may be easier to understand and write then the previous while loop version.

```
public static String removeSpaces1(String str){
        String ans = ""
        for(int i = 0; i < str.length(); i++){
                String letter = str.substring(i, i+1);
                if(!letter.equals(" "))
                        ans += letter;
        }
        return ans;
}
```

Palindrome

Given a string, return whether the string is a palindrome(reads the same forward as backwards).

```
public static boolean isPalindrome(String str){
        int len = str.length();
        for(int i = 0; i < len; i++){
                String current = str.substring(i, i + 1);
                String opposite = str.substring(len - 1 - i, len - i);
                if(!current.equals(opposite))
                        return false;
        }
        return true;
}
```

Can we make the code slightly more efficient?

Improved Palindrome

We can go just to the middle of the String.

```java
public static boolean isPalindrome(String str){
        int len = str.length();
        for(int i = 0; i < len/2; i++){
                String current = str.substring(i, i + 1);
                String opposite = str.substring(len - 1 - i, len - i);
                if(!current.equals(opposite))
                        return false;
        }
        return true;
}
```

Lab 1

On replit classroom, please do For Loops: 19 – 33 (numbers 6, 32, 33 are optional)

Lab 2

Write a static method named `printTwoDigit` that accepts an integer n as a parameter and that prints a series of n randomly generated numbers. The method should use Math.random() to select numbers in the range of 10 to 19 inclusive where each number is equally likely to be chosen.

After displaying each number that was produced, the method should indicate whether the number 13 was ever selected ("we saw a 13!") or not ("no 13 was seen."). You may assume that the value of n passed is at least 0.

You should an output similar to below. (see next slide)

Lab 2

Call	`printTwoDigit(4);`	`printTwoDigit(7);`
Output	next = 12 next = 10 next = 16 next = 11 no 13 was seen.	next = 12 next = 19 next = 12 next = 13 next = 11 next = 16 next = 13 we saw a 13!

References

1) CPJava Website
2) CPJava Google Classroom
3) CPJava repl.it Classroom
4) Runestone CSAwesome BUSHSCHOOL_CPJAVA Course
5) Building Java Programs: A Back to Basics Approach by Stuart Reges and Marty Stepp