FCDS

Programming I

Lecture 5: Switch Statement, Loops (Part I)

The `switch` statement

```
switch (integer expression) {
    case value1:
        statement(s);
        break;
    case value2:
        statement(s);
        break;
    …
    default: // if it isn't one of the above values
        statement(s);
        break;
}
```

- An alternative to the `if/else` statement
  - must be used on integral types (e.g. `int`, `char`, `long`)
  - ***break*** means exit the switch statement and continue on with the rest of the program.

Example 1

```
Scanner console = new Scanner(System.in);

System.out.print("In what place did you finish the race? ");
int place = console.nextInt();

switch (place) {
  case 1:
    System.out.println("You won the gold medal!!!");
    break;
  case 2:
    System.out.println("You earned a silver medal!");
    break;
  case 3:
    System.out.println("You got a bronze medal.");
    break;
  default:
    System.out.println("You did not win a medal. Sorry.");
    break;
}
```

Example 2

```
Scanner console = new Scanner(System.in);
System.out.print("Please enter your letter grade:");
char grade = console.next().charAt(0);

switch (grade) {
  case 'A':
   System.out.println("Excellent!!");
   break;
  case 'B':
   System.out.println("Very Good!");
   break;
  case 'C':
   System.out.println("Good!");
   break;
  case 'F':
   System.out.println("Failed");
   break;
  default:
   System.out.println("Wrong Input");
   break;
}
```

Example 3

Write a program to ask the user for the brightness of a light bulb (in Watts), and print out the expected lifetime:

Brightness	Lifetime in hours
25	2500
40, 60	1000
75, 100	750
otherwise	Wrong Input

Example 3 - answer

```
Scanner console = new Scanner(System.in);

System.out.print("Please enter the bulb brightness: ");
int bright = console.nextInt();

switch (bright) {
  case 25:
    System.out.println("Expected Lifetime is 2500 hours");
    break;
  case 40:
  case 60:
    System.out.println("Expected Lifetime is 1000 hours");
    break;
  case 75:
  case 100:
    System.out.println("Expected Lifetime is 750 hours");
    break;
  default:
   System.out.println("Wrong Input!");
   break;
}
```

Loop (Iteration) Statements

- ***Loop statements*** allow repeatedly executing a statement or a sequence of statements one or more times as long as some condition remains true.

- There are three loop statements in Java
  - ***for*** loop statement
  - ***while*** loop statement
  - ***do-while*** loop statement

Two Types of Loops

***count-controlled* loops**

repeat a specified number of times

sentinel-controlled (indefinite) loops

some condition within the loop body changes and this causes the repeating to stop

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) {
    statement;
    statement;
    ...
    statement;
}
```

header

body

- Perform **initialization** once.
- Repeat the following:
  - Check if the **test** is true. If not, stop.
  - Execute the **statements**.
  - 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
!=	not equal to
==	equal to

Loop Example

```
for  (int i = 1; i <= 3; i++){
  System.out.println("I am so smart "+i)
}
```

i ?

Loop Example

```
for (int i = 1 ; i <= 3;  i++ ){
System.out.println("I am so smart "+i);
}
```

OUTPUT

i 1

Loop Example

```
for  (int i = 1; i <= 3; i++){
    System.out.println("I am so smart "+i);
}
```

OUTPUT

i 1

Loop Example

true

```
for  (int i = 1; i <= 3; i++){
 System.out.println("I am so smart "+i);
}
```

OUTPUT

i 1

Loop Example

```
for (int i = 1; i <= 3; i++){
 System.out.println("I am so smart "+i);
}
```

OUTPUT

```
I am so smart 1
```

i 2

Loop Example

```
for (int i = 1; i <= 3; i++){
 System.out.println("I am so smart "+i);

}
```

OUTPUT

```
I am so smart 1
```

i 2

Loop Example

true

```
for  (int i = 1; i <= 3; i++){

 System.out.println("I am so smart "+i);

}
```

OUTPUT

```
I am so smart 1
```

i 2

Loop Example

```
for (int i = 1; i <= 3; i++){
 System.out.println("I am so smart "+i);
}
```

OUTPUT

```
I am so smart 1
I am so smart 2
```

i 3

Loop Example

```
for (int i = 1; i <= 3; i++){
 System.out.println("I am so smart "+i);

}
```

OUTPUT

```
I am so smart 1
I am so smart 2
```

i 3

Loop Example

true

```
for  (int i = 1; i <= 3; i++){

 System.out.println("I am so smart "+i);

}
```

OUTPUT

```
I am so smart 1

I am so smart 2
```

i 3

Loop Example

```
for (int i = 1; i <= 3; i++){
 System.out.println("I am so smart "+i);
}
```

OUTPUT

```
I am so smart 1
I am so smart 2
I am so smart 3
```

i 4

Loop Example

```
for (int i = 1; i <= 3; i++){
 System.out.println("I am so smart "+i);

}
```

OUTPUT

```
I am so smart 1
I am so smart 2
I am so smart 3
```

i 4

Loop Example

false

```
for  (int i = 1; i <= 3; i++){

 System.out.println("I am so smart "+i);

}
```

OUTPUT

```
I am so smart 1

I am so smart 2

I am so smart 3
```

i 4

Loop Example

false

```
for  (int i = 1; i <= 3; i++){

 System.out.println("I am so smart "+i);

}
```

When the loop control condition is evaluated and has value false, the loop is said to be “satisfied” and control passes to the statement following the For statement.

The output

```
I am so smart 1

I am so smart 2

I am so smart 3
```

Repetition over a range

```
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"

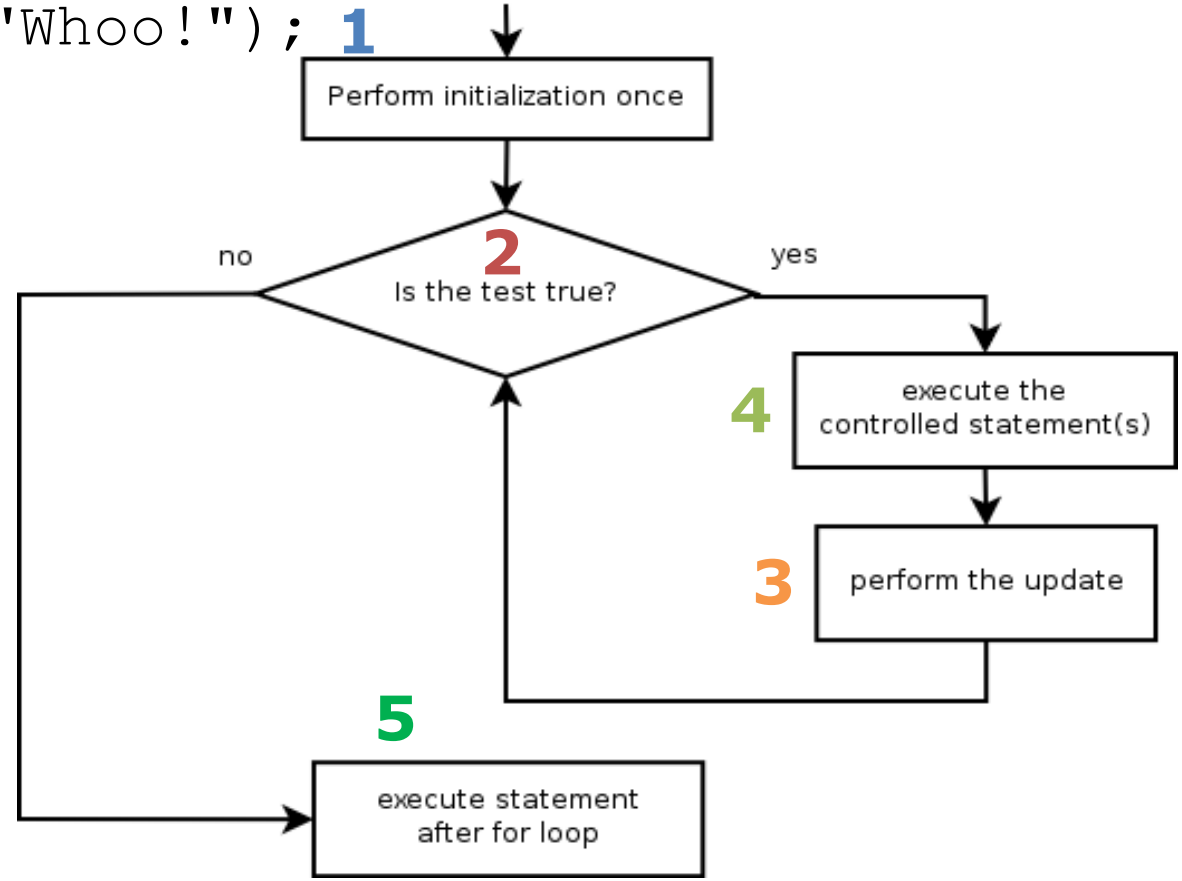
- The `for` loop does exactly that!

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

– "For each integer **i** from 1 through 6, print ..."

Loop walkthrough

```
            1          2       3
for (int i = 1; i <= 4; i++) {
 4   System.out.println(i + " squared = " + (i * i));
}
5  System.out.println("Whoo!");
```



Output:

```
1 squared = 1
2 squared = 4
3 squared = 9
4 squared = 16
Whoo!
```

Multi-line loop body

```
System.out.println("+----+");
for (int i = 1; i <= 3; i++) {
    System.out.println("\\    /");
    System.out.println("/    \\");
}
System.out.println("+----+");
```

– Output:

```
+----+
\    /
/    \
\    /
/    \
\    /
/    \
+----+
```

First iteration i=1

Second iteration i=2

Third iteration i=3

Fourth iteration i=4 **4<=3? no then stop iterating**

Expressions for counter

```
int highTemp = 5;
for (int i = -3; i <= highTemp / 2; i++) {
    System.out.println(i * 1.8 + 32);
}
```

– Output:

```
26.6
28.4
30.2
32.0
33.8
35.6
```

1st iteration $i = -3 \leq 2$, then compute $(-3 \times 1.8 + 32)$ then print

2nd iteration $i = -2 \leq 2$, then compute $(-2 \times 1.8 + 32)$ then print

3rd iteration $i = -1 \leq 2$, then compute $(-1 \times 1.8 + 32)$ then print

4th iteration $i = 0 \leq 2$, then compute $(0 \times 1.8 + 32)$ then print

5th iteration $i = +1 \leq 2$, then compute $(+1 \times 1.8 + 32)$ then print

6th iteration $i = +2 \leq 2$, then compute $(+2 \times 1.8 + 32)$ then print

7th iteration $i = +3 \leq 2$, no then exit the loop

Counting down

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

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

  - Output:

```
Count Down 10, 9, 8, 7, 6, 5, 4, 3, 2, 1,
The end.
```

Other For Loop Examples

- Increment may be greater than 1:

```
for (i=1; i<=100; i+=2)
```

  – This counts from 1 to 100 in step of 2.

  (1 3 5 7 9 11 … 101)

- Increment may be negative:

```
for (i=100; i>=5; i-=5)
```

  – This counts from 100 to 5 in steps of 5

  (100 95 90 85 … 0)

Values assigned to *i*

Infinite loop

- *Infinite Loop*: A loop that never ends.
  - Generally, you want to *avoid* these!
  - There are special cases, however, when you do want to create infinite loops on purpose.
- *Examples:*
  - *for (counter=0; counter <=10; counter--)*
  - *for (counter=50; counter >=10; counter++)*

In both examples the loops will never stop since the value of counter will never reach 10

Cumulative algorithms

Adding many numbers

- How would you find the sum of all integers from 1-1000?

```
// This may require a lot of typing

int sum = 1 + 2 + 3 + 4 + ... ;
System.out.println("The sum is " + sum);
```

- What if we want the sum from 1 - 1,000,000? Or the sum up to any maximum?
  - How can we generalize the above code?

Cumulative sum loop

```
int sum = 0;
for (int i = 1; i <= 1000; i++) {
    sum = sum + i;
}
System.out.println("The sum is " + sum);
```

Cumulative sum variable

- **cumulative sum**: A variable that keeps a sum in progress and is updated repeatedly until summing is finished.
  - The ***sum*** in the above code is an attempt at a cumulative sum.
  - Cumulative sum variables must be declared *outside* the loops that update them, so that they will still exist after the loop.

Cumulative product

- This cumulative idea can be used with other operators:

```
int product = 1;
for (int i = 1; i <= 20; i++) {
    product = product * 2;
}
System.out.println("2 ^ 20 = " + product);
```

Cumulative product variable

  – This example finds the value of 2^{20}. How would we make the base and exponent adjustable?

Example - Factorial

```
public class Factorial {

    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.print("Enter an integer to get its factorial:");
        int n = console.nextInt();
        int factorial = 1;
        System.out.print("Factorial of " + n + " is equal to ");
        for(int i = 1; i <= n; i++) {
            factorial = factorial * i;
        }
        System.out.println(factorial);
    }
}
```

Cumulative product variable

`Scanner` and cumulative sum

- We can do a cumulative sum of user input:

```
Scanner console = new Scanner(System.in);
int sum = 0;
for (int i = 1; i <= 100; i++) {
    System.out.print("Type a number: ");
    sum = sum + console.nextInt();
}
System.out.println("The sum is " + sum);
```

`if/else` in the loop

- Write a program `countFactors` that counts the number of factors of an integer.
  - `For example:` the count of factors of 24 is 8 because 1, 2, 3, 4, 6, 8, 12, and 24 are factors of 24.
- Solution:

```
// Returns how many factors the given number has.
Scanner console = new Scanner(System.in);
System.out.print("Please enter an integer: ");
int n = console.nextInt();
int count = 0;
for (int i = 1; i <= n; i++) {
    if (n % i == 0) {
        count++;  // i is a factor of number
    }
}
System.out.print("The number of factors is: " + count);
```

Counter variable

Text Processing

Remember

Type `char`

- **`char`** : A primitive type representing single characters.
  - A `String` is stored internally as an array of `char`

```
String s = "Ali G.";
```

index	*0*	*1*	*2*	*3*	*4*	*5*
value	`'A'`	`'l'`	`'i'`	`' '`	`'G'`	`'.'`

- The `char`s in a `String` can be accessed using the `charAt` method.
  - accepts an `int` index parameter and returns the `char` at that index

```
String food = "cookie";
char firstLetter = food.charAt(0);    // 'c'
```

The `for loop` and `charAt` method

- You can use a `for` loop to print or examine each character.

```
String major = "CMPS";
for (int i = 0; i < major.length(); i++){ // output:
    char c = major.charAt(i);              // C
    System.out.println(c);                 // M
}                                          // P
                                           // S
```

- **Another example**

```
// prints the alphabet
for (char c = 'a'; c <= 'z'; c++) {
    System.out.print(c);
}
```

Comparing `char` values

- You can compare `chars` with ==, !=, and other operators:

```
/* count the number of occurrences of letter 'i' in
the string "Beirut Arab University" */

    String Univ = "Beirut Arab Univeristy";
    int count = 0;
    for (int j = 0; j < Univ.length(); j++){
        if(Univ.charAt(j)=='i') count++;
     }
   System.out.println(count);

// Output: 3
```