CMSC 150
INTRODUCTION TO COMPUTING

ACKNOWLEDGEMENT: THESE SLIDES ARE ADAPTED FROM SLIDES PROVIDED WITH INTRODUCTION TO PROGRAMMING IN JAVA: AN INTERDISCIPLINARY APPROACH, SEDGEWICK AND WAYNE (PEARSON ADDISON-WESLEY 2007)

LECTURE 3

• IF, WHILE, FOR
• SCOPE
• NESTING
• OTHER CONTROL FLOW STATEMENTS
CONTROL FLOW

• Control flow.
  • Sequence of statements that are actually executed in a program.
  • Conditionals and loops: enable us to choreograph control flow.

straight-line control flow

control flow with conditionals and loops
CONDITIONALS
LETS SAY YOU WANT TO BE A POLL WORKER FOR A CAUCUS OR PRIMARY

• You have to sort people by their political party

• If a person is republican they take one ballot, otherwise they are democrat and have a different ballot

• How could we automate this?
IF STATEMENT

- **if** statement. A common branching structure.
  - Evaluate a boolean expression.
  - If **true**, execute some statements.
  - If **false**, execute other statements.

```java
if (boolean expression) {
    statement T;
} else {
    statement F;
}
```

- **boolean expression**
- **true**
- **false**
- **statement T**
- **statement F**

Can be any sequence of statements.
IF STATEMENT

• Example of control flow with if

```java
if (x < 0) x = -x;
```

```java
if (x > y) max = x;
else max = y;
```

```java
int t = x;
x = y;
y = t;
```
IF STATEMENT

• Ex. Take different action depending on value of variable.

1. public class Flip {
2.     public static void main(String[] args) {
3.         if (Math.random() < 0.5) System.out.println("Heads");
4.         else System.out.println("Tails");
5.     }
6. }

% java Flip
Heads
% java Flip
Heads
% java Flip
Tails
% java Flip
Heads
### IF STATEMENT EXAMPLES

<table>
<thead>
<tr>
<th>absolute value</th>
<th>if ( (x &lt; 0) ) ( x = -x; )</th>
</tr>
</thead>
</table>
| **put \( x \) and \( y \)** into sorted order | if \( (x > y) \)  
  ```c
  { 
  int t = x; 
  x = y; 
  y = t; 
  }
  ``` |
| maximum of \( x \) and \( y \)          | if \( (x > y) \) max = x; 
  else max = y; |
| **error check for division operation** | if \( (\text{den} == 0) \) System.out.println("Division by zero"); 
  else System.out.println("Quotient = " + num/den); |
| **error check for quadratic formula**   | double discriminant = \( b^2 - 4.0*c \); 
  if (discriminant < 0.0)  
  ```c
  { 
  System.out.println("No real roots"); 
  } 
  ``` 
  else 
  ```c
  { 
  System.out.println((-b + Math.sqrt(discriminant))/2.0)); 
  System.out.println((-b - Math.sqrt(discriminant))/2.0); 
  }
  ``` |
ACTIVITY – WITH A PARTNER

• Write an algorithm using if and else statements to output three numbers a, b, c in sorted order. You don’t have to write valid Java. This is just called pseudocode, i.e., code-like statements
  • Example pseudocode vs Java
    Output a
    vs
    System.out.println(a);
  • Or
    \[ a \leftarrow 0 \]
    vs
    int a = 0;
ELSE IF STATEMENTS

• Can allow more than two options with else-if statement
• Ex. Pay a certain tax rate depending on income level.

1. `double` rate;
2. `if` (income < 47450) rate = 0.22;
3. `else if` (income < 114650) rate = 0.25;
4. `else if` (income < 174700) rate = 0.28;
5. `else if` (income < 311950) rate = 0.33;
6. `else` rate = 0.35;

5 mutually exclusive alternatives...

<table>
<thead>
<tr>
<th>Income</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 – 47,450</td>
<td>22%</td>
</tr>
<tr>
<td>47,450 – 114,650</td>
<td>25%</td>
</tr>
<tr>
<td>114,650 – 174,700</td>
<td>28%</td>
</tr>
<tr>
<td>174,700 – 311,950</td>
<td>33%</td>
</tr>
<tr>
<td>311,950 – ∞</td>
<td>35%</td>
</tr>
</tbody>
</table>
**ELSE IF STATEMENTS**

- Why didn’t we use this program?

1. `double` rate = 0.35;
2. `if` (income < 47450) rate = 0.22;
3. `if` (income < 114650) rate = 0.25;
4. `if` (income < 174700) rate = 0.28;
5. `if` (income < 311950) rate = 0.33;

<table>
<thead>
<tr>
<th>Income</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 – 47,450</td>
<td>22%</td>
</tr>
<tr>
<td>47,450 – 114,650</td>
<td>25%</td>
</tr>
<tr>
<td>114,650 – 174,700</td>
<td>28%</td>
</tr>
<tr>
<td>174,700 – 311,950</td>
<td>33%</td>
</tr>
<tr>
<td>311,950 – ∞</td>
<td>35%</td>
</tr>
</tbody>
</table>

5 mutually exclusive alternatives...
ACTIVITY

• Could we rework our algorithm to sort 3 numbers with else-if statements to make it more clear?
EXERCISE – WITH A PARTNER

• Write a program that takes three integer command-line arguments and prints equal if all three are equal, and not equal otherwise.

• Add statements to your first program which ensure three and only three arguments were given to the program. Output a good error message so that “exception:ArrayIndexOutOfBounds” doesn’t occur and you know what went wrong in your program. Hint: Use args.length to see how many arguments there are.

• Fix this Java excerpt

```java
if (x = b && x != a)
    DoSomething();
```
THE WHILE LOOP
**WHILE LOOP**

- **while** loop. A common repetition structure.
  - Evaluate a boolean expression.
  - If true, execute some statements.
  - Repeat.

```plaintext
while (boolean expression) {
    statement 1;
    statement 2;
}
```
WHILE LOOP: POWERS OF TWO

• Ex. Print powers of 2 that are \( \leq 2^N \).
  • Increment \( i \) from 0 to \( N \).
  • Double \( v \) each time.

1. `int i = 0;`
2. `int v = 1;`
3. `while (i <= N) {`
4. `System.out.println(i + " " + v);`
5. `i = i + 1;`
6. `v = 2 * v;`
7. `}`

<table>
<thead>
<tr>
<th>i</th>
<th>v</th>
<th>i &lt;= N</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>true</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>true</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>true</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>true</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>true</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>true</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>false</td>
</tr>
</tbody>
</table>

\( N = 6 \)
ACTIVITY – WHILE LOOP

• What is wrong with the following code?
• What happens?
• Fix it and explain what the code outputs

1. int i = 0;
2. while (i <= N)
3.   System.out.println(i);
4.   i = i + 5;
ACTIVITY – WHILE LOOP

• Write an algorithm (in pseudocode) to compute the number of digits an integer has.
  • Example: input – 34567 output – 5

• Bonus: modify your algorithm to compute the number of “digits” for any base, e.g., binary, octal, or hexadecimal
EXAMPLE: IMPLEMENTING MATH.SQRT()

• Newton-Raphson method to compute $\sqrt{c}$:
  • Initialize $t_0 = c$
  • Repeat-until $t_i = c/t_i$, up to desired precision:
    - set $t_{i+1}$ to be the average of $t_i$ and $c/t_i$

<table>
<thead>
<tr>
<th>$t_i$</th>
<th>$t_{i+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_0$</td>
<td>2.0</td>
</tr>
<tr>
<td>$t_1$</td>
<td>$\frac{1}{2} (t_0 + \frac{c}{t_0}) = 1.5$</td>
</tr>
<tr>
<td>$t_2$</td>
<td>$\frac{1}{2} (t_1 + \frac{c}{t_1}) = 1.4166666666666665$</td>
</tr>
<tr>
<td>$t_3$</td>
<td>$\frac{1}{2} (t_2 + \frac{c}{t_2}) = 1.4142156862745097$</td>
</tr>
<tr>
<td>$t_4$</td>
<td>$\frac{1}{2} (t_3 + \frac{c}{t_3}) = 1.4142135623746899$</td>
</tr>
<tr>
<td>$t_5$</td>
<td>$\frac{1}{2} (t_4 + \frac{c}{t_4}) = 1.414213562373095$</td>
</tr>
</tbody>
</table>

computing the square root of 2
EXAMPLE: IMPLEMENTING MATH.SQRT()

- Newton-Raphson method to compute $\sqrt{c}$:
  - Initialize $t_0 = c$
  - Repeat-until $t_i = c/t_i$, up to desired precision: set $t_{i+1}$ to be the average of $t_i$ and $c/t_i$

```
1. public class Sqrt {
2.    public static void main(String[] args) {
3.        double epsilon = 1e-15;
4.        double c = Double.parseDouble(args[0]);
5.        double t = c;
6.        while (Math.abs(t - c/t) > t*epsilon) {
7.            t = (c/t + t) / 2.0;
8.        }
9.        System.out.println(t);
10.    }
11. }
```

% java Sqrt 2.0
1.414213562373095
ACTIVITY – WHILE LOOP

• Reverse guessing game – Write a program which takes as input $N$ and a number $g$. Generate random numbers in the range $[1, N]$ until $g$ is generated. Output the number of guesses the computer took.

• Bonus
  • Protect your program input with if statements.
  • Allow the computer to repeat the guessing process for $g$ 10 times. Average the number of guesses taken.
QUESTION DAY

• This is your chance to ask about all things java. Consider it a review and clarification time! I will explain anything you want to the best of my ability.
EXAMPLES — WITH A PARTNER

• What does are the values of $n$ and $m$ after this:
  ```
  int n = 1234567;
  int m = 0;
  while(n != 0) {
    m = (10*m) + (n % 10);
    n /= 10;
  }
  ```

• Show the trace of the program at each step
EXERCISE – WITH A PARTNER

• Random walk. You begin standing at the center of a disk of radius $r$. At each time-step you pick a random direction with respect to the $x$-axis and take a step of 1 meter. How many steps did it take you to fall off?
  • Start at $(x, y) = (0, 0)$; *YES DECIMAL PLACES ARE ALLOWED*
  • Randomly generate theta $\theta \in [0, 2\pi)$
  • Then your new position $(x, y) = (x + \cos(\theta), y + \sin(\theta))$
  • Bonus: Bias the random walk so that the random direction isn’t 100% random.
  • Bonus: Lets say after falling off your disk you fall on another disk, for $N$ disks. Each time you fall, you land at a random position $(r \cdot \cos(\theta), r \cdot \sin(\theta))$ where $\theta \in [0, 2\pi)$ and begin again. How many steps did it take?

• Start by planning your algorithm. Then implement it.
• This question has applications to simulating cellular and molecular systems.
THE FOR LOOP

```c
#include <stdio.h>
int main(void)
{
    int count;
    for (count = 1; count <= 500; count++)
        printf("I will not throw paper airplanes in class.\n");
    return 0;
}
```
FOR LOOPS

• **for loop.** Another common repetition structure.
  • Execute initialization statement.
  • Evaluate a boolean expression.
  • If true, execute some statements.
  • And then the increment statement.
  • Repeat.

```java
for (init; boolean expression; increment) {
  statement 1;
  statement 2;
}
```
ANATOMY OF A FOR LOOP

• Q. What does it print?

```java
int v = 1;
for (int i = 0; i <= N; i++)
{
    System.out.println(i + " " + v);
    v = 2*v;
}
```

- **int v = 1;**
  - initialize another variable in a separate statement
- **for (int i = 0; i <= N; i++)**
  - declare and initialize a loop control variable
  - loop continuation condition
- **System.out.println(i + " " + v);**
  - body
- **v = 2*v;**
  - increment
- **shorthand for i = i + 1**
## LOOP EXAMPLES

<table>
<thead>
<tr>
<th>Description</th>
<th>Code Example</th>
</tr>
</thead>
</table>
| print largest power of two less than or equal to $N$ | int $v = 1$;  
while ($v \leq N/2$)  
$v = 2*v$;  
System.out.println($v$); |
| compute a finite sum $(1 + 2 + \ldots + N)$      | int $sum = 0$;  
for (int $i = 1$; $i \leq N$; $i++$)  
$sum += i$;  
System.out.println($sum$); |
| compute a finite product $(N! = 1 \times 2 \times \ldots \times N)$ | int $product = 1$;  
for (int $i = 1$; $i \leq N$; $i++$)  
$product *= i$;  
System.out.println($product$); |
| print a table of function values                 | for (int $i = 0$; $i \leq N$; $i++$)  
System.out.println($i + " 	" + 2*Math.PI*i/N$); |
PRACTICE

• Table 1: Write a for loop to output all numbers between integers $a$ and $b$

• Table 2: Write a for loop to output all command line arguments. Recall: `args.length` gives the number of command line arguments

• Table 3: Write a for loop to output the multiples of an integer $a$ up to $N$

• Table 4: Write a for loop to output all the even numbers from 100 to 999 in reverse order.
NESTING
NESTING

• In control flow, nesting is where you place a control structure inside of another

• Example: 2 for loops to print a multiplication table

1. for(int i = 0; i < 10; ++i) {
2.       for(int j = 0; j < 10; ++j)
3.       System.out.printf("%d*%d = %2d\t", i, j, i*j);
4.   System.out.println();
5. }

NESTED IF STATEMENTS

- Use nested if statements to handle multiple alternatives.

```java
1. if (income < 47450) rate = 0.22;
2. else {
3.   if (income < 114650) rate = 0.25;
4.   else {
5.     if (income < 174700) rate = 0.28;
6.     else {
7.       if (income < 311950) rate = 0.33;
8.       else rate = 0.35;
9.     }  
10.   }  
11. }  
```

- Or use the shorthand:

```java
1. if (income < 47450) rate = 0.22;
2. else if (income < 114650) rate = 0.25;
3. else if (income < 174700) rate = 0.28;
4. else if (income < 311950) rate = 0.33;
5. else rate = 0.35;
```
MONTE CARLO SIMULATION
GAMBLER'S RUIN

• Gambler's ruin. Gambler starts with $stake and places $1 fair bets until going broke or reaching $goal.
  • What are the chances of winning?
  • How many bets will it take?

• One approach. Monte Carlo simulation.
  • Flip digital coins and see what happens.
  • Repeat and compute statistics.
public class Gambler {
    public static void main(String[] args) {
        int stake = Integer.parseInt(args[0]), goal = Integer.parseInt(args[1]); T = Integer.parseInt(args[2]);
        int wins = 0;
        // repeat experiment T times
        for (int t = 0; t < T; t++) {
            // do one gambler's ruin experiment
            int cash = stake;
            while (cash > 0 && cash < goal) {
                // flip coin and update
                if (Math.random() < 0.5) cash++;
                else cash--;
            }
            if (cash == goal) wins++;
        }
        System.out.println(wins + " wins of " + T);
    }
}
OTHER CONTROL FLOW STATEMENTS
DO-WHILE LOOP

• **do-while loop.** Guaranteed to execute at least once!
  • Execute sequence of statements.
  • Check loop-continuation condition.
  • Repeat.

```java
do {
    statement 1;
    statement 2;
} while (boolean expression);
```
EXAMPLE: DO-WHILE

• Average a set of numbers

1. Scanner s = new Scanner(System.in);
2. double sum = 0, number = 0;
3. do {
4. System.out.print("Enter a number (0 to quit): ");
5. number = s.nextDouble();
6. sum += number;
7. } while(number != 0);
8. System.out.println("Sum: " + sum);
COMPARISON OF LOOPS

• for loop – used when you know how many times to execute or each iteration has a natural increment

• while loop – used to execute 0 or more times. Pre-condition check.

• do-while loop – used to execute 1 or more time. Post-condition check.
OTHER HELPFUL STATEMENTS FOR LOOPS

- **break** – immediately exit the loop. Do not continue executing any more of the loop:

```java
while(true) {
    if(q-key-is-pressed()) //quit the game
        break;
    Game-loop();
}
```

- **continue** – immediately skip to the end of the body of the loop, i.e., start next iteration (checking the condition):

```java
for(int i = 0; i < 10; ++i) {
    if(isPrime(i)) //OCD against prime numbers
        continue;
    HandleNotPrimes();
}
```
MULTIPLE CONDITIONS WITH SWITCH

- **Switch statement.** Allows multiple alternatives just like with if-else.
  - Expression must be of type `char`, `byte`, `int`, `String`, etc. But no floating point values!
  - `default` is like `else`
  - `break` exits switch block

```
switch(expression) {
  case firstValue: statements; break;
  case secondValue: statements; break;
  default: statements;
}
```

**Example**

1. `char keyPressed;
2. switch(keyPressed) {
3.   case ‘w’: MoveUp(); break;
4.   case ‘a’: MoveLeft(); break;
5.   case ‘s’: MoveDown(); break;
6.   case ‘d’: MoveRight(); break;
7.   default: StandStill();
8. }
```
CONTROL FLOW SUMMARY

• Control flow.
  • Sequence of statements that are actually executed in a program.
  • Conditionals and loops: enable us to choreograph the control flow.

<table>
<thead>
<tr>
<th>Control Flow</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Straight-line programs</td>
<td>All statements are executed in the order given</td>
<td></td>
</tr>
<tr>
<td>Conditionals</td>
<td>Certain statements are executed depending on the values of certain variables</td>
<td>if; if-else; switch</td>
</tr>
<tr>
<td>Loops</td>
<td>Certain statements are executed repeatedly until certain conditions are met</td>
<td>while; for; do-while</td>
</tr>
</tbody>
</table>