1.3 Conditionals and Loops

Conditionals

Control Flow

If Statement

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

A Foundation for Programming

Any program you might want to write:
- Objects
- Functions and modules
- Graphics, sound, and image I/O
- Arrays
- Conditionals and loops
- Math
- Text I/O
- Primitive data types
- Assignment statements

A Foundation for Programming
If Statement

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

```java
public class Flip {
    public static void main(String[] args) {
        if (Math.random() < 0.5) System.out.println("Heads");
        else System.out.println("Tails");
    }
}
```

Ex. Take different action depending on value of variable.

If Statement Examples

<table>
<thead>
<tr>
<th>absolute value</th>
<th>if (x &lt; 0) x = -x;</th>
</tr>
</thead>
<tbody>
<tr>
<td>just x and y</td>
<td>if (x &gt; y) { int t = x; x = y; y = t; }</td>
</tr>
<tr>
<td>sort order</td>
<td>if (x &gt; y) max = x; else max = y;</td>
</tr>
<tr>
<td>maximum of x</td>
<td>if (den != 0) System.out.println(&quot;Division by zero&quot;); else System.out.println(&quot;No real roots&quot;);</td>
</tr>
</tbody>
</table>
| x and y        | if (den != 0) double discriminant = b*b - 4*a*c;
|                | if (discriminant < 0) System.out.println("No real roots");
|                | else {
|                | System.out.println(c-b + Math.sqrt(discriminant))/2/b);
|                | System.out.println(c-b - Math.sqrt(discriminant))/2/b); |
| error check    | if (den == 0) System.out.println("Division by zero"); else System.out.println("No real roots"); |
| for quadratic  | |
| formula        | |

The For Loop

The for loop. A 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?
A.
For Loop: Powers of Two

Ex. Print powers of 2 that are ≤ 2⁸.
- Increment \( i \) from 0 to \( N \).
- Double \( v \) each time.

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

```
i v i <= N
0 1 true
1 2 true
2 4 true
3 8 true
4 16 true
5 32 true
6 64 true
7 128 false
```

N = 6

For Loops: Subdivisions of a Ruler

Create subdivision of a ruler.
- Initialize \( r \) to " ".
- For each value \( i \) from 1 to \( N \):
  - sandwich two copies of \( r \) on either side of \( i \).

```
public class RulerN {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        String ruler = " ";
        for (int i = 1; i <= N; i++) {
            ruler = ruler + i + ruler;
        }
        System.out.println(ruler);
    }
}
```

```
i 1 2 1
2 1 2 1
3 1 2 1 3 1 2 1
4 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
5 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1 5 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
N = 6
```

Observation. Loops can produce a huge amount of output!

The While Loop

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

```
while (boolean expression) {
    statement 1;
    statement 2;
}
```

```
i 1 2 1
2 1 2 1
3 1 2 1 3 1 2 1
4 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
5 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1 5 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
while (i <= N)
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
```

```
i v i <= N
0 1 true
1 2 true
2 4 true
3 8 true
4 16 true
5 32 true
6 64 true
7 128 false
```

N = 6

While Loop: Powers of Two

Ex. Print powers of 2 that are ≤ 2⁸.
- Increment \( i \) from 0 to \( N \).
- Double \( v \) each time.
**While Loop Challenge**

**Q.** Anything wrong with the following code for printing powers of 2?

```java
public class PowersOfTwo {
    public static void main(String[] args) {
        // last power of two to print
        int N = Integer.parseInt(args[0]);
        int i = 0; // loop control counter
        while (i < N) {
            System.out.print(i + 1 + " ");
            i = i + 1;
            System.out.print(i + " ");
            System.out.print(i + 1 + " ");
            i = i + 1;
        }
    }
}
```

A. Need curly braces around statements in while loop; otherwise it enters an infinite loop, printing “0 1”.

**Moment of panic. How to stop infinite loop?**

**While Loops: Square Root**

**Goal.** Implement Math.sqrt().

Newton-Raphson method to compute the square root of c:
- Initialize \( t_0 = c \).
- Repeat until \( t_i = c / t_i \) up to desired precision:
  - \( t_{i+1} \) to be the average of \( t_i \) and \( c / t_i \).

```java
import java.math.BigDecimal;

public class Sqrt {
    public static void main(String[] args) {
        double c = Double.parseDouble(args[0]);
        double t = c;
        while (Math.abs(t - c / t) > 1e-15) {
            t = (c / t + t) / 2.0;
        }
        System.out.println(t);
    }
}
```

Technical conditions: \( f(x) \) is smooth; \( t_0 \) is good estimate.

---

**Powers of Two**

```java
public class PowersOfTwo {
    public static void main(String[] args) {
        // last power of two to print
        int N = Integer.parseInt(args[0]);
        int i = 0; // loop control counter
        while (i < N) {
            System.out.print(i + 1 + " ");
            i = i + 1;
            System.out.print(i + " ");
            System.out.print(i + 1 + " ");
            i = i + 1;
        }
    }
}
```

**While Loops: Square Root**

**Goal.** Implement Math.sqrt().

Newton-Raphson method to compute the square root of c:
- Initialize \( t_0 = c \).
- Repeat until \( t_i = c / t_i \) up to desired precision:
  - \( t_{i+1} \) to be the average of \( t_i \) and \( c / t_i \).

```java
import java.math.BigDecimal;

public class Sqrt {
    public static void main(String[] args) {
        double c = Double.parseDouble(args[0]);
        double t = c;
        while (Math.abs(t - c / t) > 1e-15) {
            t = (c / t + t) / 2.0;
        }
        System.out.println(t);
    }
}
```

Technical conditions: \( f(x) \) is smooth; \( t_0 \) is good estimate.
Nested If Statements

Ex. Pay a certain tax rate depending on income level.

<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 – 35%</td>
<td></td>
</tr>
</tbody>
</table>

Graduated income tax calculation

```java
double rate;
if (income < 47450) rate = 0.22;
else if (income < 114650) rate = 0.25;
else if (income < 174700) rate = 0.28;
else if (income < 311950) rate = 0.33;
else rate = 0.35;
```

Nested If Statements

Need all those braces? Not always.

```java
if (income < 47450) rate = 0.22;
else if (income < 114650) rate = 0.25;
else if (income < 174700) rate = 0.28;
else if (income < 311950) rate = 0.33;
```

is shorthand for

```java
if (income < 47450) rate = 0.22;
else if (income < 114650) rate = 0.25;
else if (income < 174700) rate = 0.28;
else if (income < 311950) rate = 0.33;
```

but be careful when nesting if-else statements. [See Q+A on p. 75.]

Nested If Statement Challenge

Q. What's wrong with the following for income tax calculation?

```java
double rate = 0.35;
if (income < 47450) rate = 0.22;
if (income < 114650) rate = 0.25;
if (income < 174700) rate = 0.28;
if (income < 311950) rate = 0.33;
```

Wrong graduated income tax calculation
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]);
        int goal = Integer.parseInt(args[1]);
        int 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);
        }
    }
}
```

Control Flow Summary

- 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</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>

Digression: Simulation and Analysis

Fact. Probability of winning = \(\frac{\text{stake}}{\text{goal}}\).
Fact. Expected number of bets = \(\text{stake} \times \text{desired gain}\).
Ex. 20% chance of turning $500 into $2500, but expect to make one million $1 bets.

Remark. Both facts can be proved mathematically; for more complex scenarios, computer simulation is often the best (only) plan of attack.

Extra Slides
Oblivious Sorting

Sort. Read in 3 integers and rearrange them in ascending order.

```java
class Sort3 {
    public static void main(String[] args) {
        int a = Integer.parseInt(args[0]);
        int b = Integer.parseInt(args[1]);
        int c = Integer.parseInt(args[2]);

        if (b > c) { int t = b; b = c; c = t; }
        if (a > b) { int t = a; a = b; b = t; }
        if (b > c) { int t = b; b = c; c = t; }

        System.out.println(a + " " + b + " " + c);
    }
}
```

Puzzle 1. Sort 4 integers with 5 compare-exchanges.
Puzzle 2. Sort 6 integers with 12.

Do-While Loop

**Ex.** Find a point (x, y) that is uniformly distributed in unit disc.

1. Pick a random point in unit square.
2. Check if point is also in unit disc.
3. Repeat.

```java
x = 2.0 * Math.random() - 1.0;
y = 2.0 * Math.random() - 1.0;
while (x*x + y*y > 1.0);
```