Even More Options with if-else-if

We can extend if/else to allow more than two different options

```
if (ConditionOne)
  {InstructionsOne;}
else if (ConditionTwo)
  {InstructionsTwo;}
else
  {InstructionsThree;}
```

Scanner scan;
scan = new Scanner(System.in);
int netPay = scan.nextInt();
double rate = 0.0;
if (netPay < 20000)
  {rate = 0.05;}
else if (netPay < 50000)
  {rate = 0.1;}
else if (netPay < 80000)
  {rate = 0.2;}
else
  {rate = 0.5;}
double taxes = rate * netPay;

Are these the same?

```
Scanner scan;
scan = new Scanner(System.in);
int netPay = scan.nextInt();
double rate = 0.0;
if (netPay < 20000) {
  rate = 0.05;
} else if (netPay < 50000) {
  rate = 0.1;
} else if (netPay < 80000) {
  rate = 0.2;
} else {
  rate = 0.5;
}
double taxes = rate * netPay;
```

Are these the same?

```
Scanner scan;
scan = new Scanner(System.in);
int netPay = scan.nextInt();
double rate = 0.0;
if (netPay < 20000) {
  rate = 0.05;
} else if (netPay < 50000) {
  rate = 0.1;
} else if (netPay < 80000) {
  rate = 0.2;
} else {
  rate = 0.5;
}
double taxes = rate * netPay;
```
Code Blocks: Variable Scope

- The set of instructions inside the braces { } of something like an if-statement is a code block.
- The scope of a variable defines where it is "visible" and can be used.
- A variable is only visible inside the block in which it was declared, including all nested blocks.

```
if ( ConditionHolds )
{
    // Code Block
}
```

double pay = Math.random() * 50000;
if ( pay < 20000 )
{
    double rate = 5.0;
    double tax = rate * pay;
}

double rate = 0.0;
if ( pay < 20000 )
{
    rate = 5.0;
    double tax = rate * pay;
}

Won’t work: the rate variable is not visible here.

OK: here both variables are visible, since both are declared outside of the block, and can be used both inside and outside of the block.

Randomly Choosing Values

- Suppose we want to randomly choose either a Heads or Tails image of a coin.
- What do we need here?
  - Random numbers
    - We could import and use a Random object...
    - Often, simpler to use Math
    - Like Color, certain elements that we can call directly, like Math.sqrt() and Math.random()

```
import java.lang.Math;

double E = Math.E;
double PI = Math.PI;

double abs( double )
int abs( int )
double log( double )
```

Using Math.random()

- public static double random()
  - post: returns value x randomly from range 0.0 ≤ x < 1.0

- Math.random() method generates a randomly selected double between 0.0 and 0.9999999...

```
public static double random()
{
    return Math.random();
}
```

- How can we use it to get other random numbers?
  1. double 0.0 ≤ x < 10.0
     - 10 * Math.random()
  2. double 5.0 ≤ x < 15.0
     - 10 * Math.random() + 5
  3. int 5 ≤ x < 15
     - (int) (10 * Math.random() + 5)
  4. int 5 ≤ x ≤ 15
     - (int) (11 * Math.random() + 5)
  5. int 1 ≤ x ≤ 10
     - (int) (10 * Math.random() + 1)
Using Math.random()

- In general, when using Math.random() to generate some random positive integer values, we must ask ourselves two important questions:
  1. How many different values do we want?
     - This is the range, \( R \)
     - We multiply by range
  2. What is the lowest (starting) value?
     - This is the offset, \( S \)
     - We add the offset

\[ \text{(int)( } 11 \times \text{Math.random()} + 5 \text{ )} \]

- Range \( R = 11 \) gives us 11 different possible values
- Offset \( S = 5 \) values start at 5 and go up from there

The \( \text{int} \) cast at the end ensures that we end up with a whole number among the 11 possibilities: 5, 6, ..., 14, 15

This Week & Next

- Meetings next week:
  - Wednesday: in the CS Lab (16 Wing)
  - Monday/Tuesday/Friday: regular classroom
- Program 02: due 11:59 PM, Wednesday 19 February
- Reading: Chapter 4 due Noon, Friday 21 February
- Office Hours: Wing 212
  - Monday/Wednesday/Friday: 11:00 AM–12:00 PM
  - Tuesday: 3:15 PM–4:15 PM
- CS Lab & Tutor Hours: Posted on my webpage