Flexibility in Programs

- So far, our programs have been mainly one-directional
  - Given the same inputs and the same commands, the same thing is done every time it runs
- For many applications we want programs to change their behavior at times
  - Different program results occur, depending upon factors that are determined at runtime
  - That is, behavior isn’t known ahead of time, and can change while the program is running

The if (Conditional Branch) Command

- Simplest form of Java control:
  1. **ConditionHolds**: any expression with a boolean value (true/false)
  2. **Instructions**: execute if ConditionHolds is true (otherwise the code doesn’t do anything)

  ```java
  double netPay = 20000;
  double rate = 0.0;
  if (netPay < 30000) {
      rate = 0.1;
  }
  double taxes = rate * netPay;
  ```

- When Instructions is only one line, the braces are optional, and can be left off

Results from multiple runs of the same program:
- Flipping a coin: Heads!
- Flipping a coin: Tails!
- Flipping a coin: Tails!
- Flipping a coin: Tails!
- Flipping a coin: Heads!
- Flipping a coin: Tails!
- Flipping a coin: Heads!
- Flipping a coin: Heads!
Using the if Command

Scanner class has a number of non-
void methods that check input
format and return boolean values.
We can use them as the condition of
an if-statement:

```
Scanner scan;
scan = new Scanner( System.in );
int i = 0;
if ( scan.hasNextInt() )
{
    i = scan.nextInt();
}
```

If the user enters an int value,
variable i will get that value.
Otherwise, i stays 0.

Adding Options with the else Command

We often have multiple different options we sometimes want to run:
1. If ConditionHolds is true at runtime, then Instructions execute
2. If ConditionHolds is false, then OtherInstructions execute instead

Again, braces are optional when a set of instructions is only one line

```
if ( ConditionHolds )
{
    Instructions;
}
else
{
    OtherInstructions;
}
```

Order Matters!

```
Scanner scan;
scan = new Scanner( System.in );
if ( scan.hasNextDouble() )
{
    double num1 = scan.nextDouble();
    System.out.println( num1 / 2.0 );
}
else
{
    int num2 = scan.nextInt();
    System.out.println( num2 / 2 );
}
```

This is similar to the prior code, but behaves differently when run.

```
Scanner scan;
scan = new Scanner( System.in );
if ( scan.hasNextDouble() )
{
    double num1 = scan.nextDouble();
    System.out.println( num1 / 2.0 );
}
else
{
    int num2 = scan.nextInt();
    System.out.println( num2 / 2 );
}
```

If the user enters either integer or floating-point values, it will always be read as a double.
Since every numeric type can be widened to a double, hasNextDouble() returns true for any number you give it.*

(*The number cannot be too large, but otherwise, anything works.)
Relations between Java Primitives

- When using primitives (int, double, char, etc.), we can easily compare values.
- Use relational operators.
- Result a boolean value (true/false)

== equal to
!= not equal to
< less than
<= less than/equal to
> greater than
>= greater than/equal to

```java
int i = 3;
if ( i <= 4 ) {
    System.out.println( i );
}
```

```java
double n = 3.6;
double m = 4.5;
if ( n != m ) {
    System.out.println( n );
}
```

```java
String s = "Hello!";
char c1 = s.charAt( 2 );
char c2 = s.charAt( 3 );
if ( c1 == c2 ) {
    System.out.println( "Same!" );
}
```

Two Things to be Careful about

1. Don't confuse equality (==) with assignment (=)

```java
double x = 0.1
if ( x = 0.1 ) {
    System.out.println( "Lucky!" );
}
```

Error! We want == here.

2. When using non-primitive objects, like Rectangles or Ovals or Strings, do not assume that relational operators will work! (May need class-specific methods.)

```java
String s1 = "Hello!";
String s2 = new String( "Hello" );
if ( s1 == s2 ) {
    System.out.println( "Same!" );
    // WRONG!! No output seen.
}
```

```java
String s1 = "Hello!";
String s2 = new String( "Hello" );
if ( s1.equals( s2 ) ) {
    System.out.println( "Same!" );
    // RIGHT!! Output is seen.
}
```

This Week & Next

- **Meetings this week:**
  - Monday/Wednesday: regular classroom
  - Tuesday/Friday: in the CS Lab (16 Wing)

- **Program 02**: due 11:59 PM Monday 23 September
- **Reading**: Chapter 3 due 12:00 PM, Thursday 19 Sept.
- **Office Hours**: Wing 212
  - Monday/Friday: 2:15 PM–3:15 PM
  - Tuesday: 3:00 PM–4:00 PM
  - Wednesday: 12:05 PM–1:00 PM
- **CS Lab & Tutor Hours**: Posted on my webpage