Problem: What if the user enters invalid input?
- a negative starting balance?
- interest above 100%?
- interest as an integer when you are expecting a decimal?
- withdrawing an amount more than the current balance?

We ignored this in the homework assignment, but in the future we'll always want to do some error checking on any input values

Solution: (in pseudocode)

Get amount to withdraw from the user.
If amount is greater than balance,
   print an error message.
Else,
   calculate the new balance.
Print out the account's balance.

Solution: (in pseudocode)

Get interest rate from the user.
If interest rate is less than zero or greater than 100,
   print an error message.
Else,
   if interest rate is greater than 1,
      divide interest rate by 100.0;
   calculate the new balance.
Print out the account's balance.
Control statements

- Allow us to alter the flow of a program from sequential to selection-based
  - execute some statements when certain conditions are met
- Also called *conditional statements* or *selection statements*
- Examples:
  - if/else
  - switch

**if statement**

- Example:
  ```java
  int num = Math.sqrt(56);
  if (num < 7) {
      System.out.println("Less than 7");
  }
  else {
      System.out.println("Greater than 7");
  }
  ```

**Format of an if statement**

```java
if ( < some condition is true >) {
    < do something >
}
else if ( < some other condition is true >) {
    < do something else >
}
else {
    < do something else >
}
```
Notes

- Can have as many or as few else if's as we want
- Do not need else if's or else
- Each if, else if, and else's statements are enclosed in {...}

Writing conditions

- What is the format of the < some condition is true > part of the if statement?
  - if (age < 30)
  - if (num > 7)

Boolean expressions

- *Boolean expressions* are statements that evaluate to either true or false
- true and false are boolean variables in Java
  - another primitive data type
- Format:
  <operand> <relational operator> <operand>

Examples

```java
int a = 10, b = 13, c = 20;
boolean test1 = ((a+c) % 2 == 0);
boolean test2 = (a+b < c);
boolean test3 = (a+b != c);

Q: What are the values of test1, test2, test3?
```
Relational operators

-  `<`  less than
-  `<=`  less than or equal to
-  `==`  equal to
-  `!=`  not equal to
-  `>`  greater than
-  `>=`  greater than or equal to

Order of operations (revised)

1. Parenthesis
2. Unary operators
3. `*`, `/`, `%`
4. `+`, `-`
5. Relational operators (`>`, `>=`, `<`, `<=`)
6. Equality operators (`==`, `!=`)
7. Assignment (`=`)

Note

- A common mistake is to use the assignment operator when you mean to use the equality operator.
  - Example:
    ```java
    int a=12, b=10;
    if (a > b) {
        System.out.println("a is "+a+" and b is "+b);
    } // will print out "a is 12 and b is 10"
    if (a = b) {
        System.out.println("a is "+a+" and b is "+b);
    } // will print out "a is 10 and b is 10"
    ```

Back to our example...

```java
int a = 10, b = 13, c = 20;
boolean test1 = ((a+c) % 2 == 0);
boolean test2 = (a+b < c);
boolean test3 = (a+b != c);

Q: What are the values of test1, test2, test3?
- test1 = true
- test2 = false
- test3 = true
```
Using boolean expressions in if statements

- The statement in parentheses is evaluated.
- If the statement evaluates to true, execute the code within the brackets.
- If the statement evaluates to false, move on to the next conditional statement or to the next statement in the program.

Compound boolean expressions

- Sometimes we want to evaluate more complex expressions
  - if age is greater than 30 but less than 65, print out “You are an old geezer”
  - if y is greater than 50 but not equal to 75, multiply y by 2
  - if a, b, and c are all less than zero, print out “All three numbers are negative”

Boolean operators

- Format:
  <boolean expression> <boolean operator><boolean expression>
- Three operators:
  - &&:  True if both operands are true
  - ||:  True if either operand is true
  - !:  Take the opposite of the operand

Using boolean operators

- &&:  True if both operands are true
- ||:  True if either operand is true
- !:  Take the opposite of the operand
Truth table

| p   | q   | p&&q | p||q | !p  | !q  |
|-----|-----|------|------|-----|-----|
| true| true| true | true | false| false|
| true| false| false| true | false| true |
| false| true| false| true | true | false|
| false| false| false| false| true | true |