Week 2

- Variables (Naming, Declaring and Initialization) **Review**
- Conditionals (if , if-then , if-then-else , switch)
- Loops (while , do-while , for , breaks , continues, labels)

Variable Names (Identifiers)

They are for your use as a programmer, nobody using the program will ever see these directly.

A LEGAL variable name:

- Starts with letters, underscores "_", NO NUMBERS
- Contains NO spaces
- Is unique(you cannot have two variables named "temp")

Variable Types (REVIEW)

- **int**: short for integer, used for simple numbers

```
int example = 0;
int example2 = 1;
int example3 = 50207;
int example4 = -27;
```

- **String**: used for simple words and phrases

```
String example = "Hello World"
String example2 = "Mufasa"
String example3 ="These are 2 sentences. All in one String."
```

- **char**: used for single characters

```
char yourGrade = 'F';
char myGrade = 'A';
char space =' ';
```

The Good, the Bad and the Ugly: Variable Names

Good:

General

```
int scorePlayer, temp, x, i;
//brief, unique, easy to retype 50 times.
```

All Caps

```
int CAR_COST;
//Valid but this style usually saved for constant variables.
//More on those in another lesson...
```

Foreign Characters:

```
String Ž, §, £, ¿;
//most foreign characters compile fine and are valid.
```

Good (cont.)

Abbreviations

```
double currTemp, mTime, newNum;
//abbreviations are common and encouraged. curr->current,
//temp-> temperature, m->my, num->number, var->variable,
```

Enumaration (lists of similar names)

```
long scorePlayer1, scorePlayer2;
//Often good practice
//You can use numbers, just not at the start of a variable.
```

Bad:

- a+c //the plus sign is not a valid character
- testing1-2-3 //hypens, or minuses, are not valid in names.
- O'Reilly // Apostrophes are not valid.
- 9digitSeriesCode //starts with a number

Ugly:

Too long

```
int This_is_an_insanely_long_variable_name_that_just_keeps_goir
//While this is a legal name you shouldn't ever need
//to describe something so long in a variable name.
```

Poor capitalization

```
int TempNumber = 4;
//It is best to avoid starting an identifier with lowercase.
//Uppercase is reserved for objects, more on those later...
```

Vague, poor structure

```
String thething = "Hello";
// Vague. The second word should either start
//with uppercase or be separated by an underscore.
```

Ugly

Other

```
String ðŸ’©ðŸ’©ðŸ’© = "I'm poop!";
//you can use emojis but wwhhhhhyyyy
```

Variable Declaration and Initialization (Review)

Declared variables can become anything.

```
int a, b, c;          // Declares three ints, a, b, and c.
int a = 10, b = 10;   // Example of initialization
```

Initialization gives a declared variable a value.

If-Then Statements

Examples:

- IF you only drink soda THEN you will get fat.
- IF you do your homework THEN you will do well in the class. OTHERWISE you will fail!!

IF-THEN in Java is done like so:

```
int testScore = 95;

if(testScore > 80){ //IF X
        System.out.println("You are awesome!"); //THEN Y
}else{ //OTHERWISE Z
        System.out.println("Try harder next time...");
}
```

Conditional notation (EXERCISE)

```
int myFavoriteNumber = 42;
int yourFavoriteNumber = 42;

if(myFavoriteNumber = yourFavoriteNumber){
   System.out.println("We have the same favorite number!")
}
```

What is wrong with the example above? (2 problems)

```
//Fixed version
int myFavoriteNumber = 42;
int yourFavoriteNumber = 42;

if(myFavoriteNumber == yourFavoriteNumber){ //use ==, not =.
   System.out.println("We have the same favorite number!");
   //Missing semi-colon.
}
```

==(Comparative), =(Declarative)

(REVIEW)

- == and = look the same but they are not!

```
int myFavoriteNumber = 42; // "=", DECLARING a variable
int yourFavoriteNumber = 42;//"=", DECLARING another variable

if(myFavoriteNumber == yourFavoriteNumber){ //COMPARING
   System.out.println("We have the same favorite number!");
   }
```

- Side note on .equals() (This will affect future lessons)

```
String myFavoriteWord = "Hip-hip";
String yourFavoriteWord = "Hip-hip";
//We will discuss these later. "==" doesn't work for words.
//That's all you need to know about words for now.
if(myFavoriteWord.equals(yourFavoriteWord){
    System.out.println("Hooray!");
}
```

If, Else, and Else If

Multiple options can be considered

```java
boolean store_1_has_eggs = false;
boolean store_2_has_eggs = true;
boolean store_3_has_eggs= true;

if(store_1_has_eggs){
    System.out.println("You buy eggs from store 1");
}
else if(store_2_has_eggs){
    System.out.println("You buy eggs from store 1");
}
else if(store_3_has_eggs){
    System.out.println("You buy eggs from store 1");
}
```

How many stores did we buy eggs from?

If + Else if VS. Just If

- (If + Else If):
  Do X if you can. If you can't do X, do Y. If you can't do Y do Z.

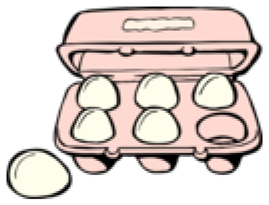
(Example: If Store 1 has eggs buy them, if it doesn't buy them from Store 2, if Store 2 doesn't have them buy from Store 3)

- Only If Statements:
  Do X if you can. Do Y if you can. Do Z if you can.

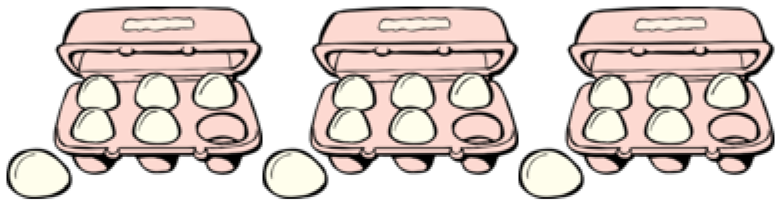
(Example: If Store 1 has eggs buy them. If Store 2 has eggs, buy them. If Store 3 has eggs, buy them.)

Remember

If + Else If Statements:



Just If Statements:



Boolean Logic and Logical Operators

- Boolean is a fancy word for "**True/False**", "**On/Off**", "**Yes/No**"
- Everything in your computer reduces to A LOT of boolean logic

```
boolean thisClassIsFun = true;
boolean londonIsInFrance = false;
int weAreCool = true; //What is wrong with this??
```

- All if-statements take ideas and convert them into checks of boolean logic.

Logical Operators:

- `&&` : AND
- `||` : OR

```java
boolean youAreHappy= true;
boolean youKnowIt= true;
if(youAreHappy && youKnowIt){
    System.out.println("Clap your hands!");
    //both need to be true
}
```

```java
boolean youKnowMyEmail = false;
boolean iKnowYourEmail = true;
if(youKnowMyEmail || iKnowYourEmail){
    System.out.println("We can communicate :)");
    //only one needs to be true
}
```

Negation, The `!` Symbol

This is used to check the opposite of something

```
if(myFavoriteNumber != yourFavoriteNumber){
   System.out.println("Our favorite numbers are different!");
}
```

With Boolean statements

```
boolean hungry = true;
boolean tired = false;
if(hungry){ //the same as: if(hungry == true)
    ...
}
if(!tired){ //the same as: if(tired == false)
    ...
}
```

More conditional symbols

```java
// Buying a pizza
boolean boughtThePizza; //Declared but not initilized
int currBalance= 500;

if(currBalance >= 10){ // '>=' means "equal or greater than"
    System.out.println("You can buy the pizza");
    boughtThePizza = true;
}else{
    System.out.println("You are too poor to eat here.");
    boughtThePizza = false;
}
```

Conditional Symbols (cont.)

```
boolean boughtThePizza = false;
int payDay = 15;
int currDate = 14;
int currBalance = 8;

if(currBalance >= 10){
    System.out.println("You can buy the pizza");
    boughtThePizza = true;
}else{
    System.out.println("You are too poor to eat here.");
    boughtThePizza = false;
}
if(currDate >= payDay){
    currBalance = currBalance + 400;
}
```

Is `boughtThePizza` true or false? Order Matters!! Let's fix it so it works. (Just rearrange the code, don't add your own)

Control Flow

```java
boolean boughtThePizza = false;
int payDay = 15;
int currDate = 14;
int currBalance = 8;

if(currDate >= payDay){
    currBalance = currBalance + 400;
}

if(currBalance >= 10){
    System.out.println("You can buy the pizza");
    boughtThePizza = true;
}else{
    System.out.println("You are too poor to eat here.");
    boughtThePizza = false;
}
```

Now that works. This is called control flow. It's has to do with the order in which you do logic. One more example:

Let's talk about order

A lot of things in life need to be done in a specific order.

Like Cooking a pizza:

```java
boolean pizzaHasSauce = true;
boolean pizzaHasCheese = false;
boolean pizzaIsBeingCooked = false;

if(pizzaHasSauce && pizzaHasCheese){ // '&&' means "and"
    System.out.println("The pizza was put in the oven!.");
    pizzaIsBeingCooked = true;
}
if(pizzaHasSauce){
    System.out.println("Cheese was added!.");
    pizzaHasCheese = true;
}
```

Fix the errors so that the logic of the control flow works and the pizzas gets cooked.

Loops

- While loops: Executes while a condition is true

```
//Prints every number from 1 to 50, starting from the top.
int number = 50;

while(number > 0){
    System.out.println(number + " cycles left");
    number-=1; //The same as "number = number - 1;"
}
```

```
//Prints every EVEN number between 0 and 10
int exampleNum = 0;

while(exampleNum <= 10){
    System.out.println(exampleNum + " is an even number");
    exampleNum+=2;
}
```

If there are loops...

There are infinite loops.

```java
int number = 50;

while(number > 0){
    System.out.println(number + " cycles left");
    //INFINITE LOOP TRIGGERED
}
```

Infinite loops are prevented by 3 things:

- Meeting conditions
- `break`
- `Continues`

Conditionals in Loops:

You can put conditionals inside or other conditionals or loops:

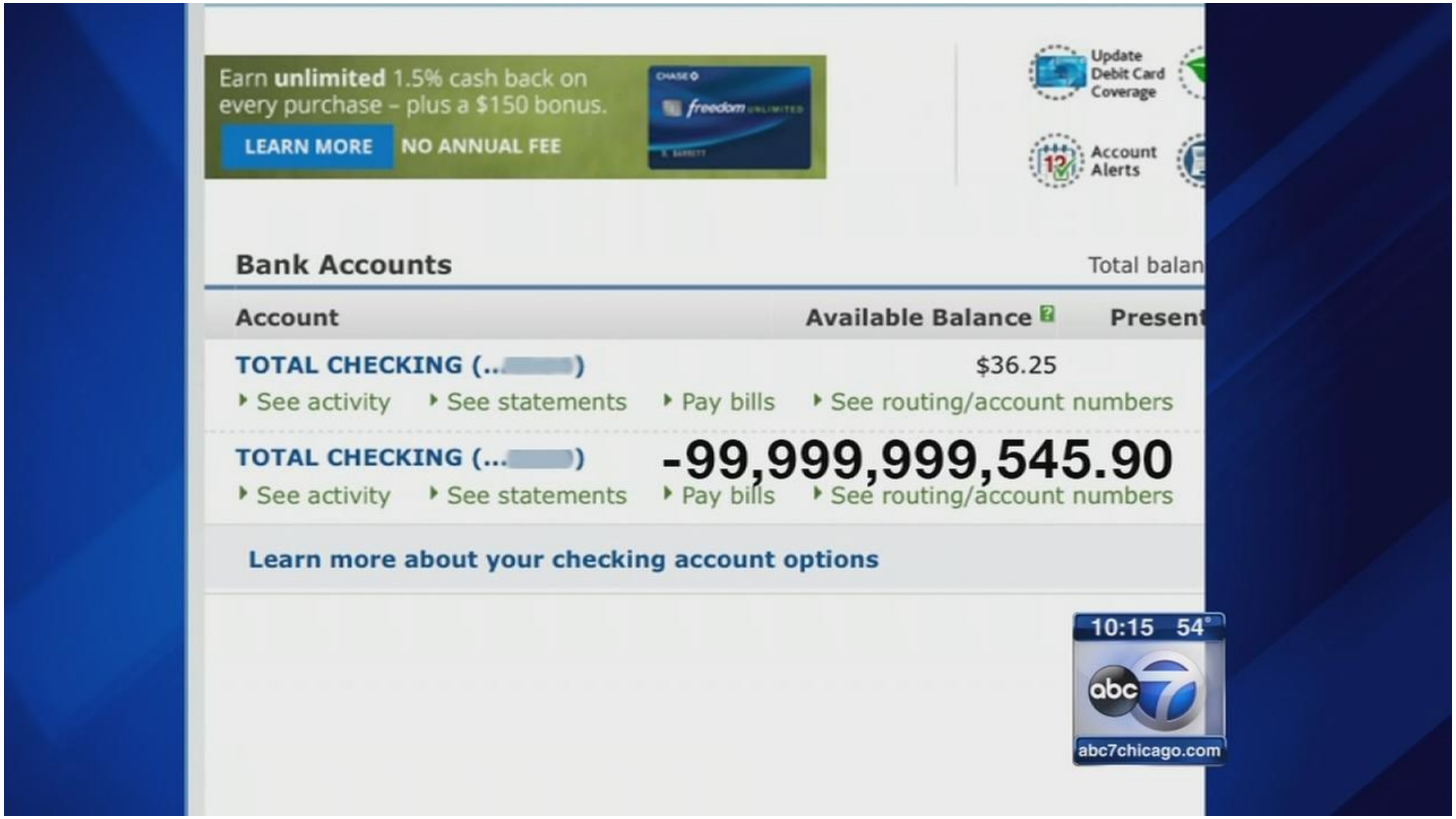
```
boolean bankAccountIsEmpty = false;
int bankAccountBalance = 85;
int burgerCount = 0;

while(!bankAccountIsEmpty){
    System.out.println("You bought a burger")
    bankAccountBalance-=5;
    burgerCount+=1;
    }
    if(bankAccountBalance <= 0){
        bankAccountIsEmpty = true;
    }

}
```

What would happen without the if-statement?

AND


Earn unlimited 1.5% cash back on every purchase – plus a $150 bonus.
LEARN MORE
NO ANNUAL FEE
Update Debit Card Coverage
Account Alerts
10:15 54°
abc7chicago.com

Bank Accounts | Total balan

Account	Available Balance	Present
TOTAL CHECKING (..[redacted])	$36.25	
TOTAL CHECKING (...[redacted])	-99,999,999,545.90	

‣ See activity ‣ See statements ‣ Pay bills ‣ See routing/account numbers

Learn more about your checking account options

Do-While Loops

Like an upside-down while loop.

Always executes at least once:

```
boolean dinnerIsReady=true;
do{
    System.out.println("Is dinner ready yet?");
} while(!dinnerIsReady)
```

How many times will this run?

```
boolean dinnerIsReady=true;
while(!dinnerIsReady){
    System.out.println("Is dinner ready yet?");
}
```

And this?

Breaks

Allows you to leave a potentially infinite loop, even if the condition is not met.

```java
//Let's say we don't want to ever change isHappy...
boolean isHappy = true;
int happyCount = 0;
while(isHappy){
    System.out.println("I'm so happy!!");
    happyCount++; //Does the same as happyCount+=1
    if(happyCount==3){
        break;
    }
}
```

Switch Statements

Switch statements are like a long list of "if-else" statements

```
char grade = 'C';

      switch(grade) {
         case 'A' :
            System.out.println("Excellent!");
            break;
         case 'B' :
         case 'C' :
            System.out.println("Well done");
            break;
         case 'D' :
            System.out.println("You passed");
         case 'F' :
            System.out.println("Better try again");
            break;
         default : //Note here
            System.out.println("Invalid grade");
      }
      System.out.println("Your grade is " + grade);
```

Switch Statements (cont.)

- **case 'value'** : forms most parts of the switch statement.
- **default:** is a special part of switch statements.
- **break;** an optional element to step out of the switch statement

Switch Practice

Write a switch statement that takes a number or char and includes:

- at least 1 `case 'value'` clause
- at least 1 `break;`
- a `default:` clause

For Loops

For loops allow you to do something a specific number of times. It has a very specific way to write it:

```java
//start value, condition, do after each loop
for(int i = 0; i < 10; i++){
    System.out.println(i + " is the number");
}//Plug in this code, how many times does it loop?
```

- `int i` is a very common way to start a for loop, this is just some value, it can be anything.
- `i++` is the same as `i=i+1` or `i+=1` it just increments `i` by 1.

More for-loop examples

```java
//What do you think this does?
int number = 3;
int tempValue = 0;
for(int i = 0; i < number; i++){
    for(int j = 0; j < number; j++){
        tempValue++;
    }
}
System.out.println(tempValue);
```

- this is called a nested for-loops they are **very** common in programming and have a LOT of uses.

For-Loop Practice!

- That last code sample "squared" a number. (x^2)
- Can you make one to "cube" a number? (x^3)

Continues

Used to return to return to the top of a loop.

```
//This is an array, we'll discuss them in a future lesson.
//Just think of this as a series of numbers, from 10 to 50.
int [] numbers = {10, 20, 30, 40, 50};

for(int x : numbers ) { //Another way to write a for-loop
         if( x == 30 ) {
            continue;
         }
         System.out.print( x );
         System.out.print("\n");
      }
```

What is the output of this block of code?

Labeled Loops:

Very rarely used but important to understand

```
int i,j;

loop1:    for(i=1;i<=10;i++){
    System.out.println();

  loop2:    for(j=1;j<=10;j++){
    System.out.print(j + " ");

    if(j==5){
        break loop1;     //Statement 1
        }
   }
}
```

Compare this result with the previous slide

```
int i,j;

for(i=1;i<=10;i++){
   System.out.println();

   for(j=1;j<=10;j++){
       System.out.print(j + " ");

       if(j==5){
           break;         //Statement 1
       }
   }
}
```

Homework! (1/4):

Q1) With just ints and for-loops, create this:

```
1
22
333
44444
555555
4444
333
22
1
```

Homework! (2/4):

Q2) Write a switch statement with:

- at least one **break;**
- a default statement

Homework! (3/4)

Q3) Write different if-blocks that use at least once:

- Negation (**!**), **&&** (and), **||** (or)
- Multiple clauses in one statement.
- If and else.
- If, else if, and else.

Homework!(4/4)

Q4) Use loops to recreate the Fibonacci Sequence with the initial indexes of 0 and 1.

- The first 10 indexes are (0,1,1,2,3,5,8,13,21,34)
- What is index #12? #20? #50?

That's all for now!