If you only understand 6 things...

- variables
- types
- output
- input
- if statements
- loops
If you only understand 6 things...

- variables
- types
- output
- input
- if statements
- loops

Which of these things affect control flow?
Variables

A variable is the name of a memory location containing a value.
A variable has 4 parts

- name or identifier
A variable is the name of a memory location containing a value.

A variable has 4 parts

- name or identifier
- storage space (memory)
Variables

A variable is the name of a memory location containing a value.

A variable has 4 parts

- name or identifier
- storage space (memory)
- the value in that storage space
Variables

A variable is the name of a memory location containing a value.

A variable has 4 parts

- name or identifier
- storage space (memory)
- the value in that storage space
- type - the kinds of values that can be put in the storage space
Variables
Variables

What can we do with them?
Variables

What can we do with them?

- declare variables

```java
String greeting;
//declare
greeting = "Hola, Mundo";
//initialize
greeting = "Dia dhuit an domhain";
//overwrite
greeting.replace("ar", "arrrh");
//call a method
```
Variables

What can we do with them?

- declare variables
- initialize variables

```java
String greeting;
//declare
greeting = "Hola, Mundo";
//initialize
greeting = "Dia dhuit an domhain";
//overwrite
greeting.replace("ar", "arrh");
//call a method
```
Variables

What can we do with them?

- declare variables
- initialize variables
- overwrite variables

String greeting;
//declare
greeting = "Hola, Mundo!";
//initialize
greeting = "Dia dhuit an domhain!";
//overwrite
greeting.replace("ar", "arrh!");
//call a method
What can we do with them?

- declare variables
- initialize variables
- overwrite variables
- ask a variable to perform actions (if the variable stores an object)
Variables

What can we do with them?

- declare variables
- initialize variables
- overwrite variables
- ask a variable to perform actions (if the variable stores an object)

```java
String greeting; //declare
greeting = "Hola, Mundo"; //initialize
greeting = "Dia dhuit an domhain"; //overwrite
greeting.replace("ar", "arrrrh"); //call a method
```
Equals is the assignment operator. Dot “.” is a separator to access an instance variable or method of the object stored by the variable.

```java
String greeting = "Hola, Mundo";
greeting = "Dia dhuit an domhain";
greeting.replace("ar", "arrrrh");
```
Variables

When to use a variable?
To remember something for later.
As a “scratch pad”.
The first computers were humans with paper.
Variables

What is in scope? What is the name of the action being performed on the variable?

```java
String greeting = "Hola, Mundo";
greeting = "Dia dhuit an domhain";
int x;
x = 7;
x = x+3;
int y = 3*x+4;
```
Type

What *type* is: an announcement that a variable will be of a certain type.

That is, the variable will have certain properties and allow certain behaviors.
Type

For example: you can check to see if an int is larger than another int, but not a String. The characters of a string can be replaced, but this makes no sense for an int.
Type

Type is a guarantee.
Does this cause an error?

```java
int iAmAString = 4;
```
Type

Type is a guarantee.
Does this cause an error?

```java
int iAmAString = 4;
```

What about this?

```java
int iAmAString = "Hello World";
```
Type

Variables have a type so the processor knows what to expect when it looks at the memory location that the variable points to.
Type

```java
int x = 4;
int[] xs = new int[4];
double y = 7.2;
String name = "Neal";
Vehicle car = new Vehicle();
boolean isTrue = true;
boolean isFalse = 7 > 8;
```
Type

When to use type?
Type

When to use type? Always

Type is mandatory when declaring variables, specifying the return type of a method, or specifying the type of method arguments.

Type is also useful for casting a variable to a new type.
Output

A way of communicating with the user. Communication from the processor to the user.
Writing to a memory location that shows up on the screen.
Output

```java
String greeting = "Hello, World";
System.out.println(greeting);
```

System is an object and out in System.out is a...?
String greeting = "Hello, World";
System.out.println(greeting);

System is an object and out in System.out is a...? out is also an object. out is an object stored as an instance variable in System.
String greeting = "Hello, World";
System.out.println(greeting);

System is an object and out in System.out is an object. println is a...?
String greeting = "Hello, World";
System.out.println(greeting);

System is an object and out in System.out is an object.
println is a...? method, an action that out can perform.
String greeting = "Hello, World";
System.out.println(greeting);

System is an object and out in System.out is an object.
println is a method, an action that out can perform.
greeting is a...?
System is an object and `out` in `System.out` is an object.

`println` is a method, an action that `out` can perform.

`greeting` is a...? `greeting` is a `String` variable that is passed to `println` as an argument.
Output

What it’s good for:

▶ to see otherwise hidden values in memory locations.
▶ to report the results of the program.
Output

String greeting = "Hello, World";
System.out.println(greeting);
System.out.print(greeting);

What’s the difference between println and print?
What’s the difference between println and print?
println puts a line break after greeting. print does not.
String greeting = "Hello, World";
int foo = 7;
System.out.println(greeting);
System.out.print(foo);

What is the name of the property that allows us to pass Strings, ints, doubles, and other types to print and println?
String greeting = "Hello, World";
int foo = 7;
System.out.println(greeting);
System.out.print(foo);

What is the name of the property that allows us to pass Strings, ints, doubles, and other types to print and println? Overloading
Input is a way for the user to interact with a program while the program is running.
The keyboard and mouse are special devices that can write data to memory that the program then reads. When the scanner calls one of its “next” methods, it is waiting on data to be placed into a special memory location by an input device.
What does the first line of the above code do?

```java
import java.util.Scanner;
Scanner in = new Scanner(System.in);
int n = in.nextInt();
```
What does the first line of the above code do?
The first line imports the package containing the Scanner class.
What does the second line of the above code do?
import java.util.Scanner;
Scanner in = new Scanner(System.in);
int n = in.nextInt();

What does the second line of the above code do?
The second line declares and initializes a variable of type scanner.
Input

```java
import java.util.Scanner;
Scanner in = new Scanner(System.in);
int n = in.nextInt();
```

What does the third line of the above code do?
What does the third line of the above code do?
The third line declares a variable of type int and initializes it with a value read in from the keyboard.
Why aren’t input and output symmetric? That is, why don’t we need to import a package for output?
Why aren’t input and output symmetric? That is, why don’t we need to import a package for output? The Scanner requires a package to be imported, but the package for output is automatically imported.
The syntax ‘new Scanner’ creates an instance of the Scanner object. System is a static object. Meaning it is always available.

```java
import java.util.Scanner;
Scanner in = new Scanner(System.in);
int n = in.nextInt();
```
Input

Input is useful whenever the user needs to interact with a program while it is running.
Some Scanner methods:

```java
import java.util.Scanner;
Scanner in = new Scanner(System.in);
boolean ready = in.hasNext();
String text = in.next();
```

Other examples can be viewed using Eclipse’s autocomplete feature.
If statement

Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;

The Road Not Taken by Robert Frost
If statement

If and else statements make it possible to conditionally execute code. If and else statements enable choice and different behaviors depending on inputs or other values in the program.
If statements are often referred to as branches. Why?
If statements are often referred to as branches. Why? because, like the diverging road, we are faced with one or more options, one or more branches to follow.
The condition of an if statement is often called a guard. Why?
The condition of an if statement is often called a guard. Why? because it guards certain code from being executed.
What is happening in this code?

```java
if (balance >= 0) {
    balance = balance +
               (INTEREST_RATE * balance) / 12;
} else {
    balance = balance - OVERDRAWN_PENALTY
}
```

Source: Java An Introduction to Problem Solving and Programming
If the balance is $\geq 0$ then the account accrues interest.
Otherwise an overdrawn penalty is applied.

Source: Java An Introduction to Problem Solving and Programming
If statement

```java
if (balance >= 0){
    balance = balance +
    (INTEREST_RATE * balance) / 12;
} else {
    balance = balance - OVERDRAWN_PENALTY
}
```

You ever get so broke the bank starts charging you money for not having enough money? - Louis CK

Source: Java An Introduction to Problem Solving and Programming
If statement

```java
if (black && white && readAllOver){
    System.out.println("Newspaper.");
} else {
    if (black && white){
        System.out.println("Zebra.");
    } else {
        System.out.println("No idea.");
    }
}
```
If statement

```java
if (black && white && readAllOver){
    System.out.println("Newspaper.");
} else if (black && white){
    System.out.println("Zebra.");
} else {
    System.out.println("No idea.");
}
```
If statement

```java
if (apple || orange) {
    if (apple && orange) {
        System.out.println("Choices, choices.");
    } else if (apple) {
        System.out.println("One a day keeps the doctor away.");
    } else {
        System.out.println("Orange you glad I didn’t say banana?");
    }
} else {
    System.out.println("No fruit for you.");
}
```
Loops are concise constructs for telling a program to repeat certain code.
Loops

Identify the following real life loops:

▶ Until you reach Yale, drive straight on Central.
Loops

Identify the following real life loops:

- Until you reach Yale, drive straight on Central. **Do-While**
Loops

Identify the following real life loops:

- Until you reach Yale, drive straight on Central. **Do-While**
- While you are waiting, read a magazine.
Loops

Identify the following real life loops:

- Until you reach Yale, drive straight on Central. **Do-While**
- While you are waiting, read a magazine. **While**
Loops

Identify the following real life loops:

- Until you reach Yale, drive straight on Central. **Do-While**
- While you are waiting, read a magazine. **While**
- Cover yours eyes for a count of 60 so we can hide.
Loops

Identify the following real life loops:

- Until you reach Yale, drive straight on Central. **Do-While**
- While you are waiting, read a magazine. **While**
- Cover yours eyes for a count of 60 so we can hide. **For**
Loops

```java
while (n > 1) {
    System.out.println(n);
    n = n - 1
}
```

```java
for (int i = 0; i < 10; i++) {
    canvas.drawOval(250 + 15 * i, 200, 10, 10);
}
```
Loops

Use loops whenever you need to repeat the same code over and over.
If you only understand 6 things...

- variables
- types
- output
- input
- if statements
- loops
Coming soon!

- string processing
- methods
- arrays
- classes and objects
- file input / output
- error handling
String processing

Strings are arrays of characters.

Think of them as lists of symbols.

String processing is a very common activity.
String processing

String processing can be finnicky, but is useful.

If we denote strings with quotes, then how do we print quotes?
String processing

String processing can be finnicky, but is useful.

If we denote strings with quotes, then how do we print quotes?

System.out.println(" \" ");
String processing

If we print quotes by escaping them with a backslash, how do we print back slashes?
String processing

If we print quotes by escaping them with a backslash, how do we print back slashes?

```java
System.out.println(" \\ ");
```
String processing

How do we print things like a “new line” for which no symbol exists?
String processing

How do we print things like a “new line” for which no symbol exists?

```
System.out.println(" \n ");
```

How do we print \n?
How do we print \n?

\n
System.out.println(" \n ");
String processing

What does this print out?

```java
System.out.println(" \\" ");
```
String processing

What does this print out?

```java
System.out.println(" \
"");
```

Trick question: it throws an error.
String processing

When I call Strings lists of characters, what do I mean?

What does this print out?

```
System.out.println("hello".charAt(1));
```
String processing

When I call Strings lists of characters, what do I mean?

What does this print out?

```java
System.out.println("hello".charAt(1));
```

It prints out “e”.
Replacing strings with other strings.
What does this print out?

```java
String normalSpeak = "A car will take you far.";
String likeAPirate =
    normalSpeak.replace("ar", "Aarrr!");
System.out.println(likeAPirate);
```
String processing

Replacing strings with other strings. What does this print out?

```java
String normalSpeak = "A car will take you far.";
String likeAPirate = normalSpeak.replace("ar", "Aarrr!");
System.out.println(likeAPirate);
```

“A cAarrr! will take you fAarrr!.”
String processing

Substrings:

What does this print out?

```java
String text = "cat in the hat";
System.out.println(text.substring(0,3));
```
String processing

Substrings:

What does this print out?

```java
String text = "cat in the hat";
System.out.println(text.substring(0, 3));
```

“cat”
String processing

Check for equality:

```java
String word = "neccessary";
boolean typo = word.equals("necessary");
```

Why don’t we use `==` as we do for numbers?
String processing

Check for equality:

```java
String word = "neccessary";
boolean typo = word.equals("necessary");
```

Why don’t we use `==` as we do for numbers? When we call `==` on object variables, we are asking if the two variables point to the same memory location.
Equality oddities

What’s the output of this code?

```java
String name = "Neal";
String sameName = "Neal";
System.out.println(name == sameName);
System.out.println(name.equals(sameName));
```

true
true

This is weird. Strings are like primitives that act like objects.
Equality oddities

What’s the output of this code?

```java
String name = "Neal";
String sameName = "Neal";
System.out.println(name == sameName);
System.out.println(name.equals(sameName));
```

true
true

This is weird. Strings are like primitives that act like objects.
Equality oddities

What’s the output of this code?

```java
Scanner in = new Scanner(System.in);
Scanner input = new Scanner(System.in);
System.out.println(in == input);
```

false because `in` and `input` reference different locations in memory.
Equality oddities

What’s the output of this code?

```java
Scanner in = new Scanner(System.in);
Scanner input = new Scanner(System.in);
System.out.println(in == input);
```

false

because in and input reference different locations in memory
Equality oddities

What’s the output of this code?

```java
Scanner in = new Scanner(System.in);
Scanner input = in;
System.out.println(in == input);
```

true because in and input reference the same locations in memory.
Equality oddities

What’s the output of this code?

```java
Scanner in = new Scanner(System.in);
Scanner input = in;
System.out.println(in == input);
```

true

because in and input reference the same locations in memory
String processing

Check for equality ignoring case:
Is this true?

```java
String name = "Neal";
System.out.println(name.equalsIgnoreCase("neal"));
```
String processing

Check for equality ignoring case:
Is this true?

```java
String name = "Neal";
System.out.println(name.equalsIgnoreCase("neal"));
```

true
String processing

Ends with certain letters:
Is this true?

String name = "Steve Buscemi";
boolean isSteveJobs = name.endsWith("Jobs");
System.out.println(isSteveJobs);
String processing

Ends with certain letters:
Is this true?

```java
String name = "Steve Buscemi";
boolean isSteveJobs = name.endsWith("Jobs");
System.out.println(isSteveJobs);
```

false
String processing

Split a string into pieces:

```java
String name = "Date,01,14,1984";
String[] csv = name.split("","");
```

What does csv contain?
String processing

Split a string into pieces:

```java
String name = "Date,01,14,1984";
String[] csv = name.split("","");
```

What does csv contain?

```
["Date", "01", "14", "1984"]
```
An array is a container object that holds a fixed number of values of a single type.

```java
int[] anArray;
// allocates memory for 10 integers
anArray = new int[10];
```

Source:

http://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html
Arrays

An array (of type book) is like a bookshelf.

byte [] anArrayOfBytes;
short [] anArrayOfShorts;
long [] anArrayOfLongs;
float [] anArrayOfFloats;
double [] anArrayOfDoubles;
boolean [] anArrayOfBooleans;
char [] anArrayOfChars;
String [] anArrayOfStrings;
Arrays

Elements

of the array can be accessed by an index.

Each element of the array is basically a variable.
Arrays

// declares an array of Strings
String[] bookshelf;
// allocates memory for 3 Strings
bookshelf = new String[3];
// initialize the elements
bookshelf[0] = "Java an Introduction";
bookshelf[1] = "The Diamond Age";
bookshelf[2] = "The Things They Carried";
Arrays

Arrays should be used when you have a bunch of related items that you need to store.
For example, grades in a class. I could create a variable with the name of each student:

```java
nealGrade = 88;
melanieGrade = 94;
kimGrade = 90;
```
Arrays

Or I could use two arrays:

```java
String[] students = ["Neal", "Melanie", "Kim"];
int[] grades = [88, 94, 90];
```

What are the advantages and disadvantages of using arrays in this case?
Classes are blueprints for objects.
Note that we’ve already seen classes:

```java
public class Helloworld
{
    public static void main(String[] args)
    {
        System.out.println("Hello, World");
    }
}
```

Sources for this material: http://taylorgroves.com/wiki/Intermediate_Java_Objects_and_Methods

http://www.voidspace.org.uk/python/articles/object_shaped_future.shtml
Classes and Objects

Classes define the behaviors and properties of objects. A particular object is called an instance of the object. For example...
A Book class defines a Book as an object with a cover sandwiching pages with writing on them. Attributes of a book include (among other things):

- author
- title
- number of pages
The Book class defines actions (methods) that can be performed on the book. These might include:

- openBook()
- closeBook()
- turnToPage(int pageNumber)
Classes and Objects

An instance of the Book object will likely have different values for its attributes than another instance of the Book object, but they have the same set of attributes.
Classes and Objects
public class Book {
    private String title;
    private String author;
    private String genre;
    private Customer renter;
    private Date2 dueDate;

    public Book(String title, String author, String genre) {
        this.title = title;
        this.author = author;
        this.genre = genre;
        renter = null;
    }

    public void checkout(Customer renter) {
        this.renter = renter;
        renter.checkOut(this);
    }

    // This code creates a date that is seven days ahead of todays date.
    dueDate = new Date2(7);
}
Classes and Objects

```java
public void checkout(Customer renter) {
    this.reenter = renter;
    renter.checkOut(this);

    // This code creates a date that
    // is seven days ahead of todays date.
    dueDate = new Date2(7);
}
```
Classes and Objects

Creating an instance of the Book object:

```java
String title = "The Things They Carried";
String author = "Tim O’Brien";
String genre = "metafiction";
Book carriedObrien = new Book(title, author, genre);
```
Classes and Objects

Objects should be used often to **encapsulate** data and how the data should be accessed and manipulated. **Object Oriented Programming** has become popular partly because it is highly intuitive to think of data in terms of objects.
A method is an operation that can be performed by, or performed on, an object. We have already seen lots of methods:

```java
public class Helloworld {
    public static void main(String[] args) {
        System.out.println("Hello, World");
    }
}
```
Methods

When a method is called, the object that implements the method is ordered to perform the task specified by the method.
Methods

Methods may take arguments and they may return a value. This is determined by the method header.

```java
public static void main(String[] args)
public static void println(char[] s)
public boolean isOverDue()
public String getTitle()
public String getAuthor()
public void checkout(Customer renter)
```

Methods can also have side effects.
Methods

Methods should be defined for a self-contained action.

```java
public boolean isOverDue()
public String getTitle()
public String getAuthor()
public void checkout(Customer renter)
```
Methods versus Functions

Both are called by name. Both may be passed data to operate on (aka parameters, aka arguments). Both optionally return data. Source:

Methods versus Functions

A method is associated with an object. A function is not.
A method has access to the object that it is associated with and therefore can modify the object. A function cannot.
A function stands alone.  

Java only uses methods, because everything in Java is associated with a class. Static methods are the closest thing Java has to functions because they can be called without instantiating an object.
File input / output

Data can be read in from a file or written out to a file. This is useful for processing large amounts of data or saving data for later use.
Files can be thought of as objects, but they are more permanent than the objects in the program and files are stored on the hard drive.
File input / output

```java
BufferedReader in = new BufferedReader(
    new FileReader(
        "K:\location\inputfile.txt"));
String text = in.readLine();
in.close();
```

Source: http://www.cs.carleton.edu/faculty/dmusican/cs117s03/iocheat.html
File input / output

```java
PrintWriter out = new PrintWriter(
    new FileWriter("K:\location\outputfile.txt"));
out.print("Hello ");
out.println("world");
out.close();
```

Source: http://www.cs.carleton.edu/faculty/dmusican/
cs117s03/iocheat.html
File input / output

File input is used for many things including processing large amounts of data. CSV files are a common type of file to be input to a program. File output is used for saving the data of a program in long term memory.
Error handling

Sometimes program errors can be anticipated and need to be handled gracefully. The try-catch construct is useful for this purpose and for debugging programs.
import java.io.FileNotFoundException;

try {
    BufferedReader in = new BufferedReader(
        new FileReader(
            "K:\\location\\inputfile.txt")
    );
}

catch (FileNotFoundException e){
    // Open a default file.
}

catch(Exception e){
    // do something else?
}
Error handling

The try-catch construct is like an if statement.
If an exception occurs then the catch block is executed immediately.
Else, no exception occurs and the catch block is not executed.
Error handling

Use try-catch for gracefully handling errors and for debugging.
Coming soon review

- string processing
- methods
- arrays
- classes and objects
- file input / output
- error handling
If you only understand 6 things...

- variables
- types
- output
- input
- if statements
- loops
BIG QUIZ

small test
Questions on the quiz

► Choose a concise and descriptive variable name
► this variable will hold values such as... what should its type be?
► Fill in the blanks in the following program... The blanks will be variable names and types.
Questions on the quiz

- If I enter ___ into this program after being prompted by ___, what will the output be?
  - This will require an understanding of inputs (of strings and ints), if statements, and boolean operators.

- Re-write a for loop as a while loop
- Re-write a while loop as a for loop
- What is the output of the following program?
  - This will require an understanding of loops, if statements, else statements, and output.