### Java Lectures

These lecture notes should be printed and you should add your own notes in the space provided.

They are intentionally incomplete so that you take the time to write down the missing bits in accordance with the results of [this study…](http://pss.sagepub.com/content/early/2014/04/22/0956797614524581.abstract) (<http://pss.sagepub.com/content/early/2014/04/22/0956797614524581.abstract>)

## Objectives:

### Be aware of the primitive data types and be able to use them in a program.

1. what is a data type
2. what is a variable : **TODO STUDENT READING** : <http://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html>
3. what is a primitive data type
4. compare a primitive type to an "object" type
5. where do you put a variable in java? Discuss scope of a variable
6. write a program demonstrating the use of data types.
7. terminology : field, instance variable.
8. INPUT/PROCESS/OUTPUT

**TODO TEACHER : go over the code in the lecture 2 package, while lecturing.**

**Variables\_01.java**

**Variables\_02.java**

**Circle\_03.java**

**Circle\_04.java**

### What is a data type

* A program consists of data and operations that you perform on that data.
* data is stored in what are called variables.
* a variable's data type determines what kind of data is contained within the variable.
* in java, variables are "statically typed" meaning you must declare the variable before you can use it
* some examples of declaring variables.  
    
  int x;  
  double y;
* Note \*\* declaring a variable for a primitive type just sets aside space for it and initializes it with a default value. Putting something in a variable is called initializing a variable.
* [Wiki](http://en.wikipedia.org/wiki/Type_system#Static_typing) quotes (java utilizes static typing)

*"A programming language is said to use static typing when type checking is performed during compile-time as opposed to run-time. "*

*"Static typing is a limited form of* [*program verification*](http://en.wikipedia.org/wiki/Program_verification) *(see* [*type safety*](http://en.wikipedia.org/wiki/Type_safety)*): accordingly, it allows many type errors to be* [*caught early*](http://en.wikipedia.org/wiki/Fail-fast) *in the development cycle."*

* example of initializing a variable  
    
  x=10;  
  y=1.13;
* you can declare and initialize at the same time

int x=10;

double y=4.9;

* **TODO TEACHER : discuss how variables are stored in memory.**

### 

### What is a primitive data type. (documentation : <http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>)

* primitive data types are simple types which have a value only. No operations are available for that value.
* java has 8 primitive data types

|  |  |
| --- | --- |
| **Data Type** | **Default Value (for fields)** |
| byte | 0 |
| short | 0 |
| int | 0 |
| long | 0L |
| float | 0.0f |
| double | 0.0d |
| char | '\u0000' |
| String (or any object) | null (Note to self, is this really a primitive data type?) |
| boolean | false |

* **TODO TEACHER : discuss objects vs primitive data types.**

### What is a variable and variable scope (<http://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html>)

### 

### See Variables\_01.java

* the place in a class where you declare a variable matters.
* there are instance variables, class variables, local variables and parameters.
* the place you put a variable affects the "visibility" or "scope" of the variable.
* when working with variables in your code, it’s important to know WHICH variables you have access to
* here is some example code to study.

package lecture2;

public class Variables {

private int instanceVariable; // this is an instance variable, every instance of the class has its OWN copy.

private static int count;// this is a class variable, every instance of the class has access to this SINGLE variable.

public static void main(String[] args) {

int localVariable;// this variable is local the the main method, you cant access this from the "callMeMethod" method.

}

public static void callMeMethod(String parameter1){

String s=parameter1;

System.out.println(s);

}

}

### Definitions (from : <http://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html>):

* Instance Variables (Non-Static Fields) Technically speaking, objects store their individual states in "non-static fields", that is, fields declared without the static keyword. Non-static fields are also known as instance variables because their values are unique to each instance of a class (to each object, in other words); the currentSpeed of one bicycle is independent from the currentSpeed of another.
* Class Variables (Static Fields)   
    
  A class variable is any field declared with the static modifier; this tells the compiler that there is exactly one copy of this variable in existence, regardless of how many times the class has been instantiated. A field defining the number of gears for a particular kind of bicycle could be marked as static since conceptually the same number of gears will apply to all instances. The code static int numGears = 6; would create such a static field. Additionally, the keyword final could be added to indicate that the number of gears will never change.
* Local Variables   
    
  Similar to how an object stores its state in fields, a method will often store its temporary state in local variables. The syntax for declaring a local variable is similar to declaring a field (for example, int count = 0;). There is no special keyword designating a variable as local; that determination comes entirely from the location in which the variable is declared — which is between the opening and closing braces of a method. As such, local variables are only visible to the methods in which they are declared; they are not accessible from the rest of the class.
* Parameters   
    
  You've already seen examples of parameters, both in the Bicycle class and in the main method of the "Hello World!" application. Recall that the signature for the main method is public static void main(String[] args). Here, the args variable is the parameter to this method. The important thing to remember is that parameters are always classified as "variables" not "fields". This applies to other parameter-accepting constructs as well (such as constructors and exception handlers) that you'll learn about later in the tutorial.

### Input/Process/Output (get the data, do something with it, output it)

* Every computer program follows this formula:
  + get the data (INPUT)
  + perform some calculation or otherwise manipulate it (PROCESS)
  + then display or otherwise indicate the results somehow (OUTPUT)
    - this could be to a display or turn on/off a motor, print, roll steel, whatever.
* we also know that the data is stored in variables.

### INPUT

* question : how does the data get into the variables. There are many ways, for now we'll look at 2 ways.
  + method 1 : hard coded data
    - we can hard code the data in the program

int radius=40;

double PI=3.14;

double area;

area=PI\*radius\*radius;

System.out.println("Area of radius "+radius+"="+area);

* method 2 : from the user via keyboard input
  + we can ask the user to enter the data from the keyboard when we run the program.
  + to get keyboard input use the Scanner class.

// input

Scanner input=new Scanner(System.in);

// prompt the user

System.out.println("Enter the radius of the circle: ");

// get the radius from the keyboard  
radius=input.nextInt();

* + inputs can be retrieved from a variety of different places.

**PROCESS**

* once data has been input, it is typically processed in some way
* data processing represents a very big part of what any program will do.
* in our example, we calculate the area of a circle therefore the processing part implies executing a mathematical formula
* java, like all programming languages support mathematical operations through the use of operators (+,-\*,,/,%) or classes in some package like Math.pow(). We'll look at mathematical operations later.

area = PI \* radius \* radius;

### OUTPUT

* output can be sent to various places and take on one of many forms
* output can be....
  + text sent to a console device
  + text or graphics sent to a GUI display window.
  + a file
  + audio in the form of speech or music
  + hard copy printout.
  + a motor turning on/off
  + etc...
* to send output to the console use System.out.println("Some text");

System.out.println("Area of radius "+ radius +"="+area);

* to send output to a GUI window   
    
  See Circle\_05.java

### Examples

1. Declaring and initializing variables

### See Variables\_02.java

/\*

\* A Simple program showing variable declarations and initializations.

\* An example is given for all primitive data types.

\* Ranges for numeric types are given in comments.

\*/

package lecture2;

public class Variables\_02 {

public static void main(String[] args) {

// all of these variables are local variables.

// declarations

// 32-bit number. Range -2,147,483,648 to 2,147,483,647

int anInteger;

int anInteger2;

int a, b, c;

// range for double (64 bit floating point #) and float (32 bit floating point #) is really big :

// Emin = -(2^(K-1) -2) and Emax = 2^(K-1)-1, inclusive, and where N and K are parameters that depend on the value set

double aDouble;

float aFloat;

// 16 bit # range = -32,768 to 32,767

short aShort;

// 8-bit # range = -128 to 127

byte aByte;

// 64-bit # range = -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

long aLong;

// 16-bit Unicode character. range = '\u0000' (or 0) to '\uffff' (or 65,535 inclusive).

// see : see http://http://en.wikipedia.org/wiki/List\_of\_Unicode\_characters

char aChar;

// is either true or false

boolean aBoolean;

// declare and initialize

int xx = 10;

int yy = 20;

// int yy = 1.2; // THIS IS AN ERROR, 1.2 is a REAL number not an INT!

int aa = 10, bb = 20, cc=30;

// initializations.

anInteger = 20;

aDouble = 34.67;

aFloat = 34.67f;

aByte = 127;

// aByte = 128;

System.out.println("byte = 127. Does it ? " + aByte);

aShort=-32768;

aShort=32767;

aLong=9223372036854775807l; // thats a BIG number!

// 0041 == 'A' see http://http://en.wikipedia.org/wiki/List\_of\_Unicode\_characters

aChar='\u0041';  
 System.out.println(aChar);

aBoolean=true;

aBoolean=false;

}

}

1. Write a program that will calculate the area of a circle. Get the inputs "radius" and "PI" from hard coded values.

See **Circle\_03.java**

1. Write a program that will calculate the area of a circle. Get the inputs from the users keyboard input using a scanner. Use the Math package for value of PI.

See **Circle\_04.java**

1. Write a program that will calculate the area of a circle. Get the inputs from the users keyboard input using a scanner. Use the Math package for value of PI. Draw a circle in a JFrame.

See **Circle\_05.java**

**Questions (some or all of these will be on the test/quiz)**

1. Define "data type".
2. What is a variable?
3. Define what "statically typed" means.
4. What happens when you declare a variable.
5. What happens when you initialize a variable.
6. Show an example where you declare an integer variable called "circumference".

int circumference;

1. Show an example where you initialize the variable circumference to 55.

circumference=55;

1. Show an example where you declare and initialize the integer variable circumference in one statement.

Int circumference=14;

1. State the 8 primitive data types (excluding String).

Int, double, float, char, boolean

1. Of the integer data types which can hold the largest number.
2. Of the real number data types which can hold the largest number.
3. Which data type can have one of only two values.
4. What’s the difference between an Object and a primitive data type?
5. What is meant by "variable scope".
6. Write a small program showing an example of each of the following.

\* instance variable

\* class variable

\* local variable

\* parameters

1. Define

Input

Process

Output.

1. State two ways of getting input.
2. Show an example for each method of obtaining input (see question 17).
3. Show the command that outputs the string "Hello World" to the console.