**Primitive Data Types Activity**

**When you think of Java variables, think of cups. Coffee cups, tea cups, giant cups that hold lots of root beer, and those big cups the popcorn comes in at a movie.**

**A variable is just a cup. A container. It holds something.**

**It has a size, and a type. In this activity you are going to look at the variables (cups) that hold primitives. Primitives are like the cups they have at the coffeehouse. If you’ve been to Starbucks, you know what what I am talking about here. They come in different sizes, and each has a name like ‘tall”, ‘grande’, and ‘venti’ (mocha latte…yum!). You might see the cups displayed on the counter so you can order appropriately.**

Tall  Grande  Venti

**And, in Java, primitives come in different sizes, and those sizes have names. When you declare a variable in Java, you must declare it with a specific type. The four containers here are for the four integer primitives in Java.**

long int short byte

**Each cup holds a value, so for Java primitives, rather than saying, “I’d like a tall French roast”, you say to the compiler, “I’d like an int variable with the number 90 please.” Except for one tiny difference…in Java you also have to give your cup a *name.* So it’s actually, “I’d like an int please, with the value of 2486, and name with variable *height.”* Each primitive variable has a fixed number of bits (cup size). The sizes for the six primitives in Java are shown below**.

     

**byte(8) short (16) int (32) long (64) float (32) double (64)**

**The eight primitive data types are boolean, char, byte, short, int, long, float, and double.**

**Here is a mnemonic device for remembering them:**

**B**e **C**areful! **B**ears **S**houldn’t **I**ngest **L**arge **F**urry **D**ogs**.**

**To explore primitive types, do the following.**

**Exercise 1**

The following program uses primitive data type *short*:

class ShortExample

{

public static void main ( String[] args )

{

short value = 32;

System.out.println("A short: " + value);

}

}

The word "number" in this program is a *variable*---a name for a section of memory that holds data using a particular data type. In this case "value" will be the name for 16 bits of main memory that uses *short* to represent an integer. (There is much more about variables in the rest of these notes.) This program puts the value 32 into "value." Then it writes out:

A short: 32

In other words, the next line of the program examines the variable and writes out what it finds.

**You Try**  
Create a file called *ShortExample.java*. Compile and run it. Check what it writes onto the screen. Now edit the program so that the 32 is changed to some other small number, say 256. Compile and run the program. Everything should be fine.

Next change the number to 35000 and try to compile and run the program. This number is too large to work with the data scheme short (in other words, it cannot be represented in 16 bits using data type *short*.) What happens?

Now edit the program (don't change the 35000) so that the data type is *int*. Compile and run the program. Is there a difference?

**Exercise 2**

The following program uses primitive data type *double*:

class DoubleExample

{

public static void main ( String[] args )

{

double value = 32;

System.out.println("A double: " + value);

}

}

Compile and run the program. Does its output (what it puts on the screen) differ from the output of the first program in the previous exercise? In this program, "value" is the name for 64 bits that uses the *double* data type to represent floating point numbers.

It is perfectly OK to use the variable name "value" for both programs. The use of names like this is a way for us to describe what we want to happen in the computer; it does not permanently reserve part of computer memory for any particular use.

Now try to "break the system." Change the "32" to a value that is too big for *double*. You may wish to use scientific notation for this.

**Exercise 3**

The following program also primitive data type *double*. This program computes and writes out the value of exp( 32 ). This is the base of natural logarithms "e" raised to the power 32. (Don't worry much about this. The point of the program is not the math but the floating point numbers.)

class DoubleCrash

{

public static void main ( String[] args )

{

double value = 32;

System.out.println("e to the power value: " + Math.exp( value ) );

}

}

Compile and run the program. Does it compile and run correctly? Now change the 32 to larger and larger numbers until something goes wrong.

**Exercise 4**

The following program uses primitive data type *char*:

class CharExample

{

public static void main ( String[] args )

{

char ch = 'A' ;

System.out.println("A char: " + ch );

}

}

The variable "ch" is 16 bits of main memory that uses a scheme for representing characters. The character 'A' has be placed in it. The program will write:

A char: A

Do the following:

* Change the 'A' into 'Z' and compile and run.
* Change the 'A' into 'AA' and try to compile the program.
* Change the 'A' into ' ' and compile and run the program.
  + Notice carefully: there is a single space between the two ' marks.
* Change the 'A' into '' and try to compile.
  + Notice carefully: there is no character between the two ' marks.
* Change the 'A' into "A" and try to compile the program.
  + (The double quotes " signify a *String*, which is something different from a *char*).

Adapted from *Head First Java* by Sierra and Bates and http://chortle.ccsu.edu/CS151/Notes/chap08/progExercises08.html