

Variables

Data Types

What is a reference?

What is a reference?

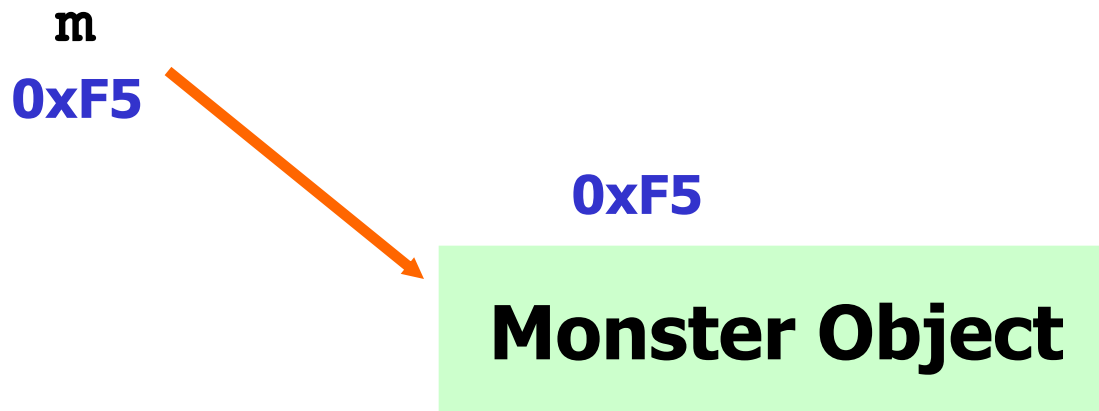
A reference variable stores the memory address of an object.

```
Monster fred = new Monster();  
Monster sally = new Monster();
```



What is a reference?

```
Monster m = new Monster();
```



m stores the address of a Monster

What is a variable?

What is a variable?

A variable is a storage location for a specified type of value.

```
int numDays = 365;  
double hTownTax = 8.25;  
char grade = 'A';
```

numDays

365


hTownTax

8.25

What is a variable?

```
int numDays = 365;
```

numDays



365

numDays stores an integer value

Identifier Names

What does identifier mean?

An identifier is used to identify something.

```
public class Triangle{ }
```

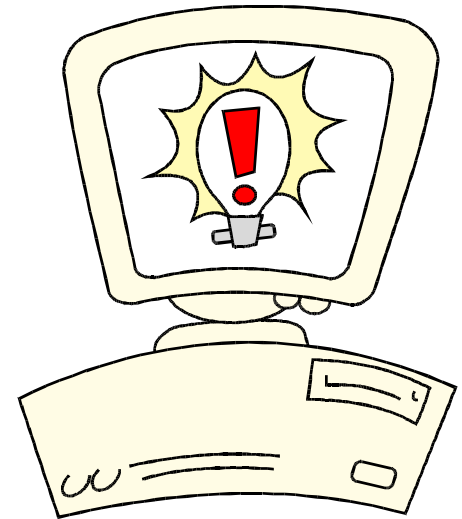
```
int width = 7;
```

Always start identifier names with letters.

Identifiers

Which of these would be legal identifiers?

1stYear
jump Up
feet2Inches
BigTriangle
SpaceInvaders



Identifier Names

Always use names that mean something.

```
double totalPay;  
class Triangle{ }
```

```
double a;  
class B{ }
```

```
//very bad  
//very bad
```

What is a keyword?

Keywords are reserved words that the language uses for a specific purpose.

**int double return void
static long break continue**

Keywords cannot be used as identifiers.

Spelling Counts

**SAM does not equal sam.
Sam does not equal sam.
Same does not equal sam.**

Case is important as is spelling.

Open

identifiers.java

Types of Variables

Data Types

byte
long

short
float

int
double

int whole
double fraction



The **type** states how much and what kind of data the variable can store.

All Data Types

data type	memory usage	min .. max
byte	8 bits	-128 to 127
short	16 bits	-32768 to 32767
int	32 bits	-2 billion to 2 billion
long	64 bits	-big to +big
float	32 bits	-big to +big
double	64 bits	-big to +big
char	16 bit unsigned	0 - 65535
reference	32 bits	n/a

It is important to know all data types and what each one can store.

Integers



Integers

```
int one = 120;  
int two = 987123;  
byte bite = 99;  
long longInt = 99234423;
```

```
System.out.println(one);  
System.out.println(two);  
System.out.println(bite);  
System.out.println(longInt);
```

OUTPUT

```
120  
987123  
99  
99234423
```

Integers

```
int one = 120.0;
```

```
System.out.println(one);
```

OUTPUT
LOP error

**Integer types can store integer values only.
Integer types cannot store fractional / decimal
values.**

**Attempting to assign fractional / decimal values to
an integer type results in a loss of precision
compile error.**

Open
integers.java
integerslop.java

Real Numbers

Fractional Values



Reals

```
double one = 99.57;  
double two = 3217;  
float three = 23.32f;
```

```
System.out.println(one);  
System.out.println(two);  
System.out.println(three);
```

OUTPUT

```
99.57  
3217.0  
23.32
```

Reals

```
double one = 120.7;  
System.out.println(one);  
one = 125;  
System.out.println(one);
```

OUTPUT

120.7

125.0

Real types can store fractional/decimal values as well as integer values.

**Open
reals.java**

characters

Characters

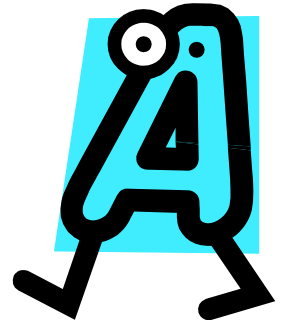
```
char let = 'A';  
char fun = 65;
```

```
char test = 'a';  
char go = 97;
```

```
char what = 48;
```

char variables are used to store a single letter.

char variables are actually integers.



Characters

char is a 16-bit unsigned int data type.

Here is a 16 bit pattern: 000000000110011

char let = 65;

ASCII VALUES YOU MUST KNOW!

'A' – 65

'a' – 97

'0' - 48

ASCII Values

'A' - 65	'B' - 66	'C' - 67	...
----------	----------	----------	-----

'a' - 97	'b' - 98	'c' - 99	...
----------	----------	----------	-----

'0' - 48	'1' - 49	'2' - 50	...
----------	----------	----------	-----

Characters

```
char alpha = 'A';  
char ascii = 65;  
char sum = 'B' + 1;
```

```
System.out.println(alpha);  
System.out.println(ascii);  
System.out.println(sum);  
System.out.println('B'+1);
```

OUTPUT

**A
A
C
67**

**Open
chars.java**

**Start work
on the labs**

Booleans

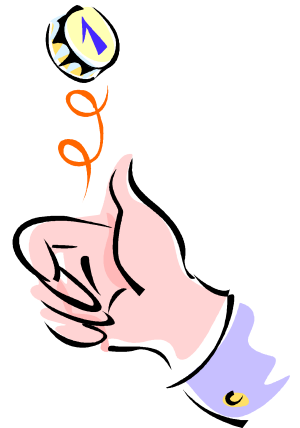
Booleans

```
boolean go = true;  
System.out.println(go);  
boolean stop = false;  
System.out.println(stop);
```

OUTPUT

true
false

A boolean type can store true or false only.



Open booleans.java

Strings

Strings

```
String dude = "hello world";  
String buddy = "whoot - \\\\[10]\\\\[10]\\\\[10]\\\\[10]\\\\[10]";
```

```
System.out.println(dude);  
System.out.println("buddy = " + buddy);
```

OUTPUT

hello world

buddy = whoot - \\\\[10]\\\\[10]\\\\[10]\\\\[10]\\\\[10]

A String type stores groups of characters.

Open strings.java

Variable Assignment

The Assignment Statement

```
receiver = 57;  
receiver = 239423;
```

In an assignment statement, the receiver is always on the left of the assignment operator (=).

Defining VS. Assigning

int **num;** ← **definition only**

int **num** = **99;** ← **definition and assignment**

num = **56;** ← **assignment only**

The Assignment Statement

```
int number = 75, bigNum=99;  
double hTownTax = 8.25;  
char bigA = 'A', littleA = 'a';  
boolean isPrime = false;  
String s = "abc";
```

```
System.out.println(number);  
System.out.println(bigNum);  
System.out.printf("%.2f\n",hTownTax);  
System.out.println(bigA);  
System.out.println(littleA);  
System.out.println(isPrime);  
System.out.println(s);
```

OUTPUT

75

99

8.25

A

a

false

abc

Open assignment.java

Data Type Ranges

All Data Types

data type	memory usage	min .. max
byte	8 bits	-128 to 127
short	16 bits	-32768 to 32767
int	32 bits	-2 billion to 2 billion
long	64 bits	-big to +big
float	32 bits	-big to +big
double	64 bits	-big to +big
char	16 bit unsigned	0 - 65535
reference	32 bits	n/a

It is important to know all data types and what each one can store.

Memory

Memory consists of bits and bytes.

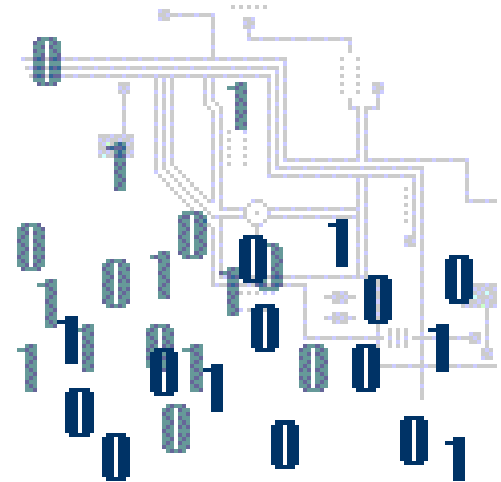


8 bits = 1001 0010 = 1 byte

16 bits = 0101 1001 0100 1001 = 2 bytes

The more bits you have the more you can store.

1 byte = 8 bits



Integer MIN and MAX

```
System.out.println(Byte.MIN_VALUE);  
System.out.println(Byte.MAX_VALUE);
```

```
System.out.println(Short.MIN_VALUE);  
System.out.println(Short.MAX_VALUE);
```

**MIN_VALUE and
MAX_VALUE are
very useful for
contest
programming.**

OUTPUT

-128
127
-32768
32767

Integer MIN and MAX

```
System.out.println(Integer.MIN_VALUE);  
System.out.println(Integer.MAX_VALUE);
```

```
System.out.println(Long.MIN_VALUE);  
System.out.println(Long.MAX_VALUE);
```

OUTPUT

-2147483648

2147483647

-9223372036854775808

9223372036854775807

Overflow Errors

```
int num = Integer.MAX_VALUE;  
num=num+1;  
System.out.println(num);  
num=num-1;  
System.out.println(num);
```

Why does adding 1 to
MAX_VALUE give you the
MIN_VALUE?

OUTPUT

```
-2147483648  
2147483647
```

Open

integersminmax.java

Real MIN and MAX

```
System.out.println(Float.MIN_VALUE);  
System.out.println(Float.MAX_VALUE);
```

```
System.out.println(Double.MIN_VALUE);  
System.out.println(Double.MAX_VALUE);
```

**MIN_VALUE and
MAX_VALUE are
very useful for
contest
programming.**

OUTPUT

```
1.4E-45  
3.4028235E38  
4.9E-324  
1.7976931348623157E308
```

Open
realsminmax.java

Character MIN and MAX

```
out.println((int)Character.MIN_VALUE);  
out.println((int)Character.MAX_VALUE);
```

```
out.println(Character.MIN_VALUE);  
out.println(Character.MAX_VALUE);
```

**MIN_VALUE and
MAX_VALUE are
very useful for
contest
programming.**

OUTPUT

0
65535
?
?

Open

charsminmax.java

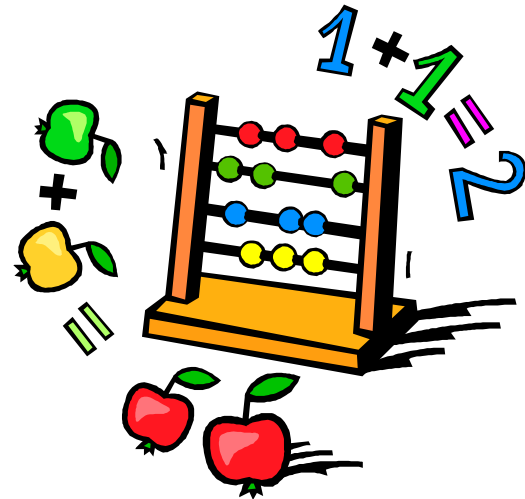
Mixing data

Mixing Data

**Java is a strong typed language.
You must pay attention to a variable's
type when assigning a value.**

```
int one=90;  
char letter= 'A';  
char let= 97;
```

```
one=letter;  
letter=let;  
one=let;
```



Mixing Data

```
int one = 90;  
double dec = 234;  
char letter = 'A';
```

```
System.out.println( one );  
one = letter;           //char to int  
System.out.println( one );
```

```
one = 'A';              //char to int  
System.out.println( one );
```

```
System.out.println( dec );  
dec = one;              //int to double  
System.out.println( dec );
```

OUTPUT

```
90  
65  
65  
234.0  
65.0
```

Data type sizes
often determine
if assignment is
legal.

32 bit == 32 bit

open
mixingdata.java

**Continue work
on the labs**

AutoBoxing

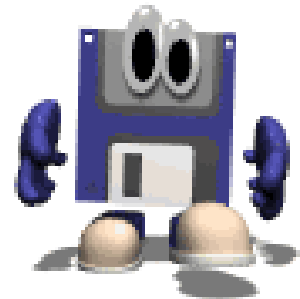
AutoUnboxing

References/Objects

In JAVA, you have 8 primitive data types.

All other variables in Java are reference variables. **References refer to objects.**

Monster m = new Monster();



References/Objects

primitive	object
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

Box/Unbox

Before Java 5 added in autoboxing and autounboxing, you had to manually wrap primitives.

```
Integer x = new Integer(98);  
int y = 56;  
x= new Integer(y);
```

Box/Unbox

Java now wraps automatically.

```
Integer numOne = 99;
```

```
Integer numTwo = new Integer(99);
```

```
=99;
```

```
=new Integer(99);
```

These two lines are equivalent.



Box/Unbox

Java now wraps automatically.

Double numOne = 99.1;

Double numTwo = new Double(99.1);

=99.1;

=new Double(99.1);

These two lines are equivalent.



Box/Unbox

Before Java 5 added in autoboxing and autounboxing, you had to manually unwrap references.

```
Integer ref = new Integer(98);  
int y = ref.intValue();
```

Box/Unbox

Java now unwraps automatically.

```
Integer num = new Integer(3);  
int prim = num.intValue();  
out.println(prim);  
prim = num;  
out.println(prim);
```

OUTPUT

3

3

```
prim=num.intValue();  
prim=num;
```

These two lines are equivalent.

Box/Unbox

```
Double dub = 9.3;  
double prim = dub;  
out.println(prim);
```

```
int num = 12;  
Integer big = num;  
out.println(big.compareTo(12));  
out.println(big.compareTo(17));  
out.println(big.compareTo(10));
```

OUTPUT

9.3

0

-1

1

Open objects.java

GUI HELP

guihelp.java