public class Types
{
    public static void main(String[] args)
    {
        int foo = 1;
        double bar = foo;
        System.out.println(foo);
        System.out.println(bar);
    }
}
Topic 07: Types

- All 8 Primitives
- Notes on Floating Point
- Notes on Characters
- Assignment in more detail
- Casting

Special Topic 2.1

2.2.5
What is a Primitive?

- Basic type
- Computer can use it directly

- Numeric types
  - Integers
  - Floating point

- Integers in disguise
  - Characters
  - Boolean
Characteristics of a Primitive

- Directly stored
  - Never use 'new' to allocate
- Only primitives have literals
  - Exception: String
- Fast & efficient
  - Low memory use
  - Rapid calculations
# All 8 Primitives - Integers

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Size</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>8 bits</td>
<td>$-128$ to $127$</td>
</tr>
<tr>
<td>short</td>
<td>16 bits</td>
<td>$-32,768$ to $32,767$</td>
</tr>
<tr>
<td>int</td>
<td>32 bits</td>
<td>$-2^{31}$ to $2^{31}-1$ (billions)</td>
</tr>
<tr>
<td>long</td>
<td>64 bits</td>
<td>$-2^{63}$ to $2^{63}-1$ (quintillions)</td>
</tr>
</tbody>
</table>
All 8 Primitives – Floating Point

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Size</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>32 bits</td>
<td>(approx) $\pm 10^{-45}$ to $10^{38}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(approx) 7 digits of accuracy</td>
</tr>
<tr>
<td>double</td>
<td>64 bits</td>
<td>(approx) $\pm 10^{-324}$ to $10^{308}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(approx) 17 digits of accuracy</td>
</tr>
<tr>
<td>Keyword</td>
<td>Size</td>
<td>Range</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>--------------------</td>
</tr>
<tr>
<td>boolean</td>
<td>1 bit</td>
<td>true, false</td>
</tr>
<tr>
<td>char</td>
<td>16 bits</td>
<td>Unicode characters</td>
</tr>
</tbody>
</table>
Literals

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>1234</td>
</tr>
<tr>
<td>long</td>
<td>1234L</td>
</tr>
<tr>
<td>float</td>
<td>1.0f</td>
</tr>
<tr>
<td>double</td>
<td>1.0</td>
</tr>
<tr>
<td>char</td>
<td>'a'</td>
</tr>
<tr>
<td>boolean</td>
<td>true</td>
</tr>
<tr>
<td></td>
<td>false</td>
</tr>
</tbody>
</table>
Topic 07: Types

- All 8 Primitives
- **Notes on Floating Point**
- Notes on Characters
- Assignment in more detail
- Casting

Special Topic 2.1

2.2.5
What is Floating Point?

- Two floating point types
  - `float` (32 bits)
  - `double` (64 bits)

- Huge numbers
  - 123,450,000,000,000,000 ...

- Tiny fractions
  - .000000000000000012345
Rounding Problems with F.P.

- Adding huge numbers to tiny numbers
  - Tiny numbers get ignored

- Adding fractions
  - Often not exactly the answer you expect

“Computerphile: Floating Point Numbers”
https://www.youtube.com/watch?v=PZRI1IfStY0
So Why Use F.P.?

- If you need fractions
- If you need huge numbers
- If you need tiny numbers

- Just remember that all floating point answers are approximate
  - Avoid `==`, use `<` instead.
Topic 07: Types

- All 8 Primitives
- Notes on Floating Point
- Notes on Characters
- Assignment in more detail
- Casting

Special Topic 2.1

2.2.5
char c1 = 'H';
char c2 = 'e';
char c3 = 'l';
char c4 = 'l';
char c5 = 'o';

System.out.print (c1);
System.out.print (c2);
System.out.print (c3);
System.out.print (c4);
System.out.println(c5);

System.out.println(""+c1+c2+c3+c4+c5);
char c1 = 'H';
char c2 = 'e';
char c3 = 'l';
char c4 = 'l';
char c5 = 'o';

System.out.print (c1);
System.out.print (c2);
System.out.print (c3);
System.out.print (c4);
System.out.println(c5);
System.out.println(""+c1+c2+c3+c4+c5);
Characters

char c1 = 'H';
char c2 = 'e';
char c3 = 'l';
char c4 = 'l';
char c5 = 'o';

Output:
Hello
Hello

System.out.print (c1);
System.out.print (c2);
System.out.print (c3);
System.out.print (c4);
System.out.println(c5);

System.out.println(""+c1+c2+c3+c4+c5);
Characters as Integers

- Characters are actually integers
  - 16 bits, **unsigned**

- Can do math, gives **int** results
- Can do comparisons
Characters

```
char c1 = 'a';
int i1 = c1+2;
int i2 = 'c'-'a';
char c2 = (char)i1;

System.out.println("c1="+c1);
System.out.println("i1="+i1);
System.out.println("i2="+i2);
System.out.println("c2="+c2);
```

Output: ????
Characters

```java
char c1 = 'a';
int i1 = c1+2;
int i2 = 'c'-'a';
char c2 = (char)i1;

System.out.println("c1="+c1);
System.out.println("i1="+i1);
System.out.println("i2="+i2);
System.out.println("c2="+c2);
```

Output:
c1=a
i1=99
i2=2
c2=c
In Class:
Can you write a method which will check to see if a single character is lowercase?
public static boolean isLowercase(char c) {
    return c >= 'a' && c <= 'z';
}

In Class:
Can you write a method which will check to see if a single character is lowercase?
In Class:
Can you write a method which will convert any lowercase letter to uppercase – and return the new character?

(Leave non-letter characters unchanged.)

Hint:
The Unicode value of 'a' is 97.
The Unicode value of 'A' is 65.
public static char toUpperCase(char c) {
    if ('a' <= c && c <= 'z') {
        return (char)(c-32);
    } else {
        return c;
    }
}
Topic 07: Types

- All 8 Primitives
- Notes on Floating Point
- Notes on Characters
- Assignment in more detail
- Casting

Special Topic 2.1

2.2.5
How Does Assignment Work?

\[ a = b; \]
How Does Assignment Work?

```plaintext
a = b;
```

![Diagram](attachment:diagram.png)
How Does Assignment Work?

\[ a = b; \]
How to Types Affect Assignment?

- Assignment works the same way with all types:
  - load
  - store
- For bigger types, we just use instructions that move more data at once.
Assignment (byte – 8 bits)

byte a;
byte b;
a = b;
Assignment (int – 32 bits)

```c
int a;
int b;
a = b;
```
Assignment (\textit{double} – 64 bits)

double \ a;  
double \ b;  
\texttt{a = b;}

\begin{itemize}
  \item \texttt{double a;}
  \item \texttt{double b;}
  \item \texttt{a = b;}
\end{itemize}
Assignment \((\text{char} \rightarrow 16 \text{ bits})\)

```c
char a;
char b;
a = b;
```
What about Mixed Types?

• If the left and right side have different types
  load
cast
store

• “Casting” converts a value from one type to another
  - Keep the value (if possible)
  - Change the way it's stored
Topic 07: Types

- All 8 Primitives
- Notes on Floating Point
- Notes on Characters
- Assignment in more detail
- Casting
Explicit Cast Example

```java
int a;
byte b;
a = (int)b;
```
Explicit Cast Example

int a;
byte b;

\[
a = (\text{int})b;
\]

Value to be converted
Explicit Cast Example

```java
int a;
byte b;
a = (int)b;
```

Destination type
Explicit Cast Example

```java
int a;
byte b;
a = (int)b;
```

Variable to store into
Explicit Cast Example

```java
int a;
byte b;
a = (int)(b*10);
```

It's legal to perform a calculation, and then cast it to another type.
Explicit Cast Example

```java
int a;
byte b;
a = ((int)b)*10;
```

It's to perform a cast as part of a larger calculation.
Assignment with an Explicit Cast

```c
int a;
byte b;
a = (int)b;
```
Assignment with an Explicit Cast

```c
int a;
byte b;
a = (int)b;
```
Assignment with an Explicit Cast

```c
int a;
byte b;
a = (int)b;
```
Assignment with an Explicit Cast

```c
int a;
byte b;
a = (int)b;
```
Dangerous & Safe Casts

byte b;
short s;
int i;
long l;

b = 13;
s = b;
i = b;
l = b;
System.out.println("b="+b);
System.out.println("s="+s);
System.out.println("i="+i);
System.out.println("l="+l);
System.out.println();

Output:
???
Dangerous & Safe Casts

```java
byte b;
short s;
int i;
long l;

b = 13;
s = b;
i = b;
l = b;

System.out.println("b+"+b);
System.out.println("s+"+s);
System.out.println("i+"+i);
System.out.println("l+"+l);
System.out.println();
```

**Output:**

```
b=13
s=13
i=13
l=13
```
Dangerous & Safe Casts

byte b;
short s;
int i;
long l;

l = 40_000L;
b = (byte)l;
s = (short)l;
i = (int)l;
System.out.println("b="+b);
System.out.println("s="+s);
System.out.println("i="+i);
System.out.println("l="+l);
System.out.println();

Output:
???
Dangerous & Safe Casts

```java
byte b;
short s;
int i;
long l;

l = 40_000L;
b = (byte)l;
s = (short)l;
i = (int)l;
System.out.println("b="+b);
System.out.println("s="+s);
System.out.println("i="+i);
System.out.println("l="+l);
System.out.println();
```

Output:
```
b=64
s=-25536
i=40000
l=40000
```
Implicit Casts

- Happen automatically, because they don't lose information (mostly)

byte → short → int → long
char → int → long
float → double

[any integer] → [any floating point]
Dangerous & Safe Casts

```java
byte b;
short s;
int i;
long l;

b = 13;
s = b;
i = b;
l = b;

System.out.println("b="+b);
System.out.println("s="+s);
System.out.println("i="+i);
System.out.println("l="+l);
System.out.println();
```

Safe casts can be implicit
Dangerous & Safe Casts

```java
byte b;
short s;
int i;
long l;

l = 40_000L;
b = (byte)l;
s = (short)l;
i = (int)l;

System.out.println("b="+b);
System.out.println("s="+s);
System.out.println("i="+i);
System.out.println("l="+l);
```

Dangerous casts must always be explicit
Casting to Floating Point

```java
public class PrintExploded {
    public static void main(String[] args) {
        System.out.print('H');
        System.out.print('e');
        System.out.print('l');
        System.out.print('l');
        System.out.print('o');
        System.out.print(' ');
        System.out.print('w');
        System.out.print('o');
        System.out.print('r');
        System.out.print('l');
        System.out.print('d');
        System.out.print('!');
        System.out.println();
    }
}
```

```java
long   long1, long2;
float float1,float2;
long1  = 2000000000;  // two billion
float1 = long1;
long2  = long1+1;
float2 = long2;
System.out.println("long1 ="+long1);
System.out.println("float1="+float1);
System.out.println();
System.out.println("long2 ="+long2);
System.out.println("float2="+float2);
System.out.println();
System.out.println("floats are equal? " + (float1 == float2));
```

Output: ???
public class PrintlnExploded {
    public static void main(String[] args) {
        System.out.print('H');
        System.out.print('e');
        System.out.print('l');
        System.out.print('l');
        System.out.print('o');
        System.out.print(' ');
        System.out.print('w');
        System.out.print('o');
        System.out.print('r');
        System.out.print('l');
        System.out.print('d');
        System.out.print('!');
        System.out.println();
    }
}

long   long1, long2;
float  float1, float2;

long1  = 2000000000;  // two billion
float1 = long1;
long2  = long1+1;
float2 = long2;

System.out.println("long1 ="+long1);
System.out.println("float1="+float1);
System.out.println();
System.out.println("long2 ="+long2);
System.out.println("float2="+float2);
System.out.println();
System.out.println("floats are equal? " +
                 (float1 == float2));
Implicit Casts

• Happen automatically, because they don't lose information *(mostly)*

byte → short → int → long
char → int → long
float → double
[any integer] → [any floating point]
(Not always precise)
Topic 07: Types

- All 8 Primitives
- Notes on Floating Point
- Notes on Characters
- Assignment in more detail
- Casting

Summary