A labs due Jan 18 Monday

Functions

methods in Scanner:

- `next(String)` converts token to String
- `nextInt()` int
- `nextLong()` long
- `nextFloat()` float
- `nextDouble()` double
- `nextBoolean()` boolean
a method is a named collection of instructions.

function main() is the entry point for program execution.

```
System.out.println("Enter ... ");
```

Instance of `PrintStream` class.

`PrintStream` belongs to `package java.lang` which is always imported by default.
also in Java, long: Math

Math contains min()

many methods in Math:
Math.exp()
Math.abs()
Math.round()
Math.random()

also constants
Math.E
Math.PI
notice some methods (such as all those in Math) are called by

```
Class Name, Method Name (argument list)
```

double x = Math.sqrt(4.0);
// x gets 2.0

other methods are called

```
Instance Of Class, Method Name (arg list)
```
```java
Scanner sc = new Scanner(System.in);
String word = sc.nextLine();
```

Instance of
Scanner class

Two categories of methods

1. Class or static methods
   ```java
   NameOfClass.NameOfMethod(...)
   ```

2. Instance methods
   ```java
   InstanceOfClass.NameOfMethod()
   ```
Recall: String contains called `concat()` (instance)

Ex.  String \( s1 = \text{"one"}; \)
     String \( s2 = \text{"two"}; \)
     String \( s3 = s1, \text{concat}(s2); \)

     instance of
     String

     (static)
     String also contains `valueOf()`

Ex.  int \( x = 6512; \)
     char \( c = \text{\'A\'}; \)

     String number = String.valueOf(x);
     // "6512"

     String letter = String.valueOf(c);
     // "A"
Conversions between types

Recall numeric types

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>8</td>
</tr>
<tr>
<td>short</td>
<td>16</td>
</tr>
<tr>
<td>int</td>
<td>32</td>
</tr>
<tr>
<td>long</td>
<td>64</td>
</tr>
<tr>
<td>float</td>
<td>32</td>
</tr>
<tr>
<td>double</td>
<td>64</td>
</tr>
</tbody>
</table>

Java has built-in arithmetic ops only for these 4 types.

Ex: type 'int':
$a = 0 \ldots 0 0 \ldots 0 0 \ldots 0 00000010$

$3 = 0 \ldots 0 0 \ldots 0 0 \ldots 0 00000011$

\text{add to set}

$s = 0 \ldots 0 0 \ldots 0 0 \ldots 0 00000101$

\[
\begin{cases}
  x = 2, 
  y = 3, 
  z, \\
  z = x + y.
\end{cases}
\]

Ex. type long

$2L = 0^0 0 0 0 0 0 0 0 0 0000010$

$3L =$

\text{addel}

$5L =$
What if we mix types?

The operand at lower type is converted to the higher type, then arithmetic takes place in higher type.

Lower vs. higher means:

\[
\text{int} < \text{long} < \text{float} < \text{double}
\]

These are called "widenings" conversions.
Ex.

```java
int x = 6;
long y;
double z;

y = x; // widening to 6L
z = y; // widening to 6.0
System.out.println(z); // output: 6.0
```

Note: Narrowing conversion can cause an error.

Ex.

```java
double z = 6.5;
int x;
x = z; // narrowing conversion
   // Syntax error
```
We must use a cast operator.

Every data type has a cast operator:

```
(type)
```

Ex: `double z = 6.5;
int x;
x = (int) z; // not an error
System.out.println(x);
// output: 6`

Ex. Round.java