Week 2 - Friday

CS 121
What did we talk about last time?

Data representation

- Binary numbers

Types

- int
- boolean
- double
- char
- String
Questions?
We've got output

- To output stuff, we just use
  ```java
  System.out.println()
  System.out.println("Flip mode is the squad!");
  System.out.println(35);
  ```

- What about input?
- Input is a little trickier
- We need to create a new object of type `Scanner`
Using Scanner

There are three parts to using **Scanner** for input

1. Include the appropriate import statement so that your program knows what a **Scanner** object is
2. Create a specific **Scanner** object with a name you choose
3. Use the object you create to read in data
Lots of people have written all kinds of useful Java code

By importing that code, we can use it to help solve our problems

To import code, you type `import` and then the name of the package or class

To import `Scanner`, type the following at the top of your program (before the `class`!)

```java
import java.util.Scanner;
```
Creating a Scanner object

- Once you have imported the Scanner class, you have to create a Scanner object.
- To do so, declare a reference of type Scanner, and use the new keyword to create a new Scanner with System.in as a parameter like so:

  ```java
  Scanner in = new Scanner(System.in);
  ```

- You can call it whatever you want, I chose to call it in.
- Doesn't make any sense? For now, that's okay.
Now that you've got a **Scanner** object, you can use it to read some data. It has a method that will read in the next piece of data that user types in, but you have to know if that data is going to be an `int`, a `double`, or a `String`. Let's say the user is going to input her age (an `int`) and you want to store it in an `int` variable called `years`. We'll use the `nextInt()` method to do so:

```java
int years;
years = in.nextInt();
```
Scanner methods

- Scanner has a lot of methods (ways to accomplish some tasks)
- For now, we're only interested in three
- These allow us to read the next `int`, the next `double`, and the next `String`, respectively:

```java
Scanner in = new Scanner(System.in);
int number = in.nextInt();
double radius = in.nextDouble();
String word = in.next();
```
import java.util.Scanner;

public class Age
{
    public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);
        System.out.println("What is your age?");
        int years;
        years = in.nextInt();
        years = years * 2;
        System.out.print("Your age doubled is ");
        System.out.println(years);
    }
}

Putting it all together
Basic Operations
Basic operations

- In **Java**, each data type has a set of basic operations you are allowed to perform.
- It’s not possible to define new operations or change how the operations behave.
- Some programming languages allow this, but **not Java**.
Operations for each type

- Today we are going to consider the basic operations for numerical types:
  - int
  - double
Operations on ints
The + Operator for int

- Use the + operator to add two ints together

```c
int a;
int b;
a = 5 + 6;    // a contains 11
b = a + 3;    // b contains 14

a + b;        // not allowed, does nothing

a = a + 1;    // a contains 12, and b?
```
Some expressions are used so often, Java gives us a short cut

- \(x = x + y\); can be written \(x += y\);
- \(x = x + 1\); can be written \(x++\);

```java
int x;

x = 6;       // x contains 6
x += 4;      // x contains 10
x++;         // x contains 11
```
The – Operator for int

- Exactly like `+` except performs subtraction

```java
int a;
int b;
a = 5 - 6;  // a contains -1
b = 3 - a;  // b contains 4
a -= 10;   // shortcut for a = a - 10;
a--;       // shortcut for a = a - 1;
```
The * Operator for int

- The * operator performs multiplication

```c
int a;
int b;
a = 5 * 6;    // a contains 30
b = a * 3;    // b contains 90
a *= 2;       // shortcut for a = a * 2;
```
The / Operator for int

- The / operator performs **integer** division
- **Not** the same as regular division

```c
int a;
int b;
a = 3;       // a contains 3
b = a / 2;   // b contains 1
a /= 2;      // shortcut for a = a / 2;
```

- The factional part is dropped, **not** rounded
The % Operator for int

- The % operator is the mod operator
- It finds the remainder after division

```c
int a;
int b;
a = 8;    // a contains 8
b = a % 5;  // b contains 3

a %= 2;    // shortcut for a = a % 2;
```

- This operator is a good way to find out if a number is even or odd
Compute the area of a rectangle

- \( \text{Area} = \text{length} \cdot \text{width} \)
Operations on doubles
The + Operator for double

- Exactly the same as + for int, except now you can have fractional parts

```c
double a;
double b;
a = 3.14159; // a contains 3.14159
b = a + 2.1; // b contains 5.24159
a += 1.6; // shortcut for a = a + 1.6;
a++; // shortcut for a = a + 1.0;
```
No surprises here
They do subtraction and multiplication

double a;
double b;
a = 3.14159; // a contains 3.14159
b = a - 2.1; // b contains 1.04159

a = b * 0.5; // a contains 0.520795
Unlike `int`, this division does have fractional parts.

```c
double a;
double b;
a = 3; // a contains 3.0
b = a / 2; // b contains 1.5
b = 3 / 2; // b contains 1.0
```

Can you explain this mystery?
The % Operator for double

- Yes, there is a % operator for doubles, but no one uses it
- So, don’t worry about it
Lab 2
Upcoming
Next time...

- Advanced mathematical operations
- Operations on *boolean* values
Reminders

- Keep reading Chapter 3 of the textbook
- Get a start on Project 1