Topic 16
Creating Correct Programs

"It is a profoundly erroneous truism, repeated by all the copybooks, and by eminent people when they are making speeches, that we should cultivate the habit of thinking about what we are doing. The precise opposite is the case. **Civilization advances by extending the number of operations which we can perform without thinking about them.** Operations of thought are like cavalry charges in a battle -they are strictly limited in number, they require fresh horses, and must only be made at decisive moments."

-**Alfred North Whitehead**

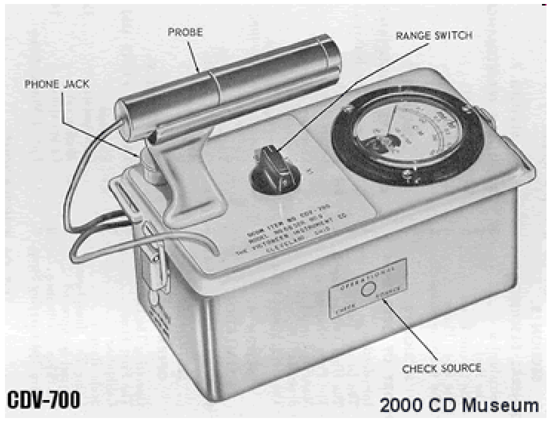
Based on slides for Building Java Programs by Reges/Stepp, found at http://faculty.washington.edu/stepp/book/

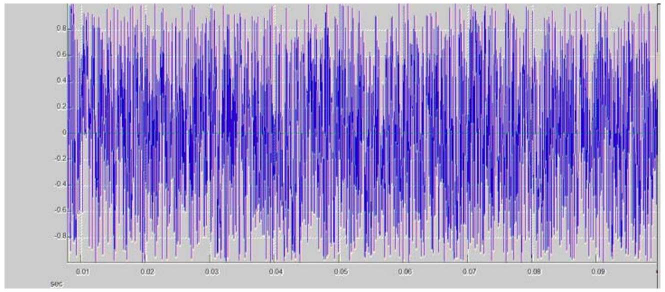
The keyword list thus far:

- Complete list of Java keywords:

abstract	default	**if**	private	this
boolean	do	implements	protected	throw
break	**double**	**import**	**public**	throws
byte	**else**	instanceof	**return**	transient
case	extends	**int**	short	try
catch	**final**	interface	**static**	**void**
char	finally	long	strictfp	volatile
class	float	native	super	**while**
const	**for**	**new**	switch	
continue	goto	package	synchronized	
assert	enum			

Generating "Random" Numbers





The Random class

- Java has a class named Random whose objects generate pseudo-random numbers.

Method name	**Description**
`nextInt()`	returns a random integer
`nextInt(`*max*`)`	returns a random integer in the range [0, max) in other words, from 0 up through one less than the max
`nextDouble()`	returns a random real number in the range [0.0, 1.0)

  - Example:

```
Random rand = new Random();
int randomNumber = rand.nextInt(10);
// randomNumber has a random value between 0 and 9
```

  - Class Random is found in the java.util package.

```
import java.util.*;
```

Pseudo Random Numbers

- What does "pseudo random" numbers mean?
- Computers don't do things ambiguously
  - despite what people think
- They have a limited number of commands
  - "Pick a random number" isn't one of them
- pseudo random numbers are generated algorithmically via mathematical operations
- They start with an initial number of see number
- If you know the seed and the algorithm you can completely predict the sequence of numbers

Pseudo Random Numbers

- Try this:

```
for(int i = 0; i < 10; i++){
    Random r = new Random(1127);
    for(int j = 0; j < 20; j++){
        System.out.print( r.nextInt() );
    }
    System.out.println();
}
```

- Try with out the initial seed.
- A whole area of computer science devoted to trying to generate seemingly random numbers

Random examples

```
Random rand = new Random();
```

- A random number between 0 and 19 inclusive:
- A random number between 1 and 10 inclusive:
- A random number between 4 and 17 inclusive:
- A random *even* number between 0 and 10 inclusive:
- A random multiple of 10 between 100 and 200 inclusive:

Random examples (answers)

```
Random rand = new Random();
```

- A random number between 0 and 19 inclusive:
```
int random1 = rand.nextInt(20);
```

- A random number between 1 and 10 inclusive:
```
int random1 = rand.nextInt(10) + 1;
```

- A random number between 4 and 17 inclusive:
```
int random1 = rand.nextInt(14) + 4;
```

- A random *even* number between 0 and 10 inclusive:
```
int random1 = rand.nextInt(6) * 2;
```

- A random multiple of 10 between 100 and 200 inclusive:
```
int random1 = rand.nextInt(11) * 10 + 100;
```

Random practice problem

- Write a multiplication tutor program. Example dialogue:

```
This program helps you learn multiplication by
Asking you 5 random multiplication questions
and counts how many you get right.

#1 of 5: 10 * 25 = 250
Correct!

#2 of 5: 72 * 12 = 864
Correct!

#3 of 5: 87 * 21 = 1741
Incorrect.  The correct answer was 1827

#4 of 5: 8 * 84 = 692
Incorrect.  The correct answer was 672

#5 of 5: 25 * 36 = 900
Correct!

You got 3 out of 5
```

The do/while Loop

The do/while loop

- Java has another kind of loop named the *do/while loop*.
  - It is almost identical to the *while loop*, except that its body statement(s) will always execute the first time, regardless of whether the condition is true.

- The do/while loop, general syntax:

```
do {
    <statement(s)> ;
} while (<condition>);
```

  - Example:

```
// roll until we get a number other than 3
Random rand = new Random();
int dice;
do {
    dice = rand.nextInt();
} while (dice == 3);
```

Creating Correct Programs and Reasoning About Programs

Assertions

- Bonus quote:
  - "As soon as we started programming, we found out to our surprise that it wasn't as easy to get programs right as we had thought. Debugging had to be discovered. I can remember the exact instant when I realized that a large part of my life from then on was going to be spent in finding mistakes in my own programs."
  - **Maurice V Wilkes**

Assertions

- **Assertion**: A declarative sentence that is either true or false
- Examples:
  - **2 + 2 equals 4**
  - **The Yankees did not play in the world series in 2006.**
  - **x > 45**
  - **It is raining.**
  - **UT beat OU last year in football.**
  - **UT volleyball will make the NCAA tourney this year.**
- Not assertions
  - **How old are you?**
  - **Take me to H.E.B.**

Assertions

- Some assertions are true or false depending on context. Which of these depend on the context?

  2 + 2 equals 4

  The Yankees did not play in the world series in 2006.

  x > 45

  It is raining.

  UT will beat OU next year.

  UT volleyball will make the NCAA tourney this year.

Assertions

- Assertions that depend on context can be evaluated if the context is provided.
  - **when x is 13, x > 45**
  - **It was raining in Round Rock, at 8 am on, October 10, 2006.**
- Many skills required to be a programmer or computer scientists
- Just a few we have seen so far
  - ability to generalize
  - create structured solutions
  - trace code
  - manage lots of details

Assertions

- Another important skill in programming and computer science is the ability "to make assertions about your programs and to understand the contexts in which those assertions will be true."

```
Scanner console = new Scanner(System.in);
System.out.print("Enter Y or N: ");
String result = console.next();
// is result equal to "Y" or "N" here?
```

Checking Input

```
Scanner console = new Scanner(System.in);
System.out.print("Enter Y or N: ");
String result = console.nextLine();
while( !result.equals("Y") && !result.equals("N")){
     System.out.print("That wasn't a Y or N. ");
     System.out.print("Enter Y or N: ");
     result = console.nextLine();
}
// is result equal to "Y" or "N" here?
```

Assertions

- **Provable Assertion**: An assertion that can be proven to be true at a particular point in program execution.
- **Program Verification**: A field of computer science that involves reasoning about the formal properties of programs to prove the correctness of a program.
  – Instead of testing.
  – A number of UTCS faculty are involved in verification research: Hunt, Lam, Shmatikov, Young, Emerson, Moore,

Examples

```
if(<test>){
     // test is always true here
}

if(<test>){
     // test is always true here
} else {
     // test is never true here
}

if(<test>){
     // test is always true here
     // other statements
     // can we assert test is true here?
}
```

Examples

```
if(<test>){
        // test is always true here

        // more statements
}
// can we assert anything about test here?

while( <test> ){
        //test is always true here
        // more statements
}
// test is never true here (unless we break...)
```

Examples

```
if( x < 0 ){
     // x < 0 is always true here
     x = -x;
}
// what about x < 0 here?
```

"Programmers use assertions naturally when writing programs. Program verification researchers try to figure out how to do this kind of reasoning in a formal, verifiable, and hopefully automated way."

Example

```
if(<test>){
     // test is always true here
}

if(<test>){
     // test is always true here
} else {
     // test is never true here
}

if(<test>){
     // test is always true here
     // other statements
     // can we assert test is true here?
}
```

Detailed Example

```
public static void printCommonPrefix(int x, int y){
    int z = 0;
    // Point A
    while( x != y ){
        // Point B
        z++;
        // Point C
        if( x > y ){
            // Point D
            x = x / 10;
        } else {
            // Point E
            y = y / 10;
        }
        // Point F
    }
    // Point G
    System.out.println("common prefix = " + x);
    System.out.println("Number of digits discarded = " + z);
}
```

Detailed Example

- Interested in assertions:
  - $x > y$
  - $x == y$
  - $z == 0$
- Are these assertions always true, never true, or sometimes true at the various labeled points in the method

Assertion example 1

```
public static int mystery(Scanner console) {
    int prev = 0;
    int count = 0;
    int next = console.nextInt();
    // Point A
    while (next != 0) {
        // Point B
        if (next == prev) {
            // Point C
            count++;
        }
        prev = next;
        next = console.nextInt();
        // Point D
    }
    // Point E
    return count;
}
```

Which of the following assertions are true at which point(s) in the code?
Choose ALWAYS, NEVER, or SOMETIMES.

`next == 0`

`prev == 0`

`next == prev`

Assertion example 2

```java
public static void mystery(int x, int y) {
    int z = 0;

    // Point A
    while (x >= y) {
        // Point B
        x -= y;

        // Point C
        z++;

        // Point D
    }

    // Point E
    System.out.println(z +
      " " + x);
}
```

Which of the following assertions are true at which point(s) in the code? Choose ALWAYS, NEVER, or SOMETIMES.

`x < y`

`x == y`

`z == 0`

Assertion example 3

```
// Assumes y >= 0, and returns x to the y power
public static int pow(int x, int y) {
    int prod = 1;
    // Point A
    while (y > 0) {
        // Point B
        if (y % 2 == 0) {
            // Point C
            x *= x;
            y /= 2;

            // Point D
        } else {
            // Point E
            prod *= x;
            y--;
            // Point F
         }
        // Point G
    }
    // Point H
    return prod;
}
```

Which of the following assertions are true at which point(s) in the code? Choose ALWAYS, NEVER, or SOMETIMES.

`y == 0`

`y % 2 == 0`

Java assert statement

- Java includes the ability to put assertions in the code as executable statements

```
assert <boolean expression>;
```