Topic 17
Assertions and Program Logic

"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

Based on slides by Marty Stepp and Stuart Reges
from http://www.buildingjavaprograms.com/

Clicker 1

- What is output by the following method?

```
public static void mysteryB(boolean b) {
    System.out.print(b + " ");
    b = (b == false);
    System.out.print(b);
}
```

A. no output due to syntax error
B. no output due to runtime error
C. not possible to predict output
D. always outputs `true false` OR `false true`
E. always outputs `true true` OR `false false`

Assertions

- **Assertion**: A declarative sentence that is either true or false
- Examples:

  2 + 2 equals 4

  The St. Louis Cardinals played in the 2011 world series

  x > 45

  It is raining.

  UT football beat OU last year.

  UT volleyball qualified for the NCAA tourney last 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 St. Louis Cardinals played in the world series this year

x > 45

It is raining.

UT football beat OU last year.

UT volleyball qualified for the NCAA tourney last 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.nextLine();
// result is 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();
}
// result is 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 and hardware to prove the correctness of the program or hardware.
  - Instead of testing.
  - A number of UTCS faculty are involved in verification and formal methods research:
    Emerson, Hunt, Lam, Moore, Young

Reasoning about assertions

- Suppose you have the following code:

```
if (x > 3) {
    // Point A
    x--;
} else {
    // Point B
    x++;
    // Point C
}
// Point D
```

- What do you know about `x`'s value at the three points?
  - Is `x > 3`? Always? Sometimes? Never?

Assertions in code

- We can make assertions about our code and ask whether they are true at various points in the code.
  - Valid answers are ALWAYS, NEVER, or SOMETIMES.

```
System.out.print("Type a nonnegative number: ");
double number = console.nextDouble();
// Point A: is number < 0.0 here?          (SOMETIMES)

while (number < 0.0) {
    // Point B: is number < 0.0 here?       (ALWAYS)
    System.out.print("Negative; try again: ");

    number = console.nextDouble();
    // Point C: is number < 0.0 here?       (SOMETIMES)
}

// Point D: is number < 0.0 here?           (NEVER)
```

Reasoning about programs

- Right after a variable is initialized, its value is known:

```
int x = 3;
// is x > 0?  ALWAYS
```

- In general you know nothing about parameters' values:

```
public static void mystery(int a, int b) {
// is a == 10?  SOMETIMES
```

Reasoning about programs

- But inside an `if`, `while`, etc., you may know something:

```
public static void mystery(int a, int b) {
    if (a < 0) {
        // is a == 10?  NEVER
        ...
    }
}
```

Assertions and loops

- At the start of a loop's body, the loop's test must be `true`:

```
while (y < 10) {
    // is y < 10?  ALWAYS
    ...
}
```

- After a loop, the loop's test must be `false`:

```
while (y < 10) {
    ...
}
// is y < 10?  NEVER
```

- Inside a loop's body, the loop's test may become `false`:

```
while (y < 10) {
    y++;
    // is y < 10?  SOMETIMES
}
```

"Sometimes"

- Things that cause a variable's value to be unknown
  (often leads to "sometimes" answers):
  - reading from a `Scanner`
  - reading a number from a `Random` object
  - initial value of a parameter in a method
- If you can reach a part of the program both with the answer being "yes" and the answer being "no", then the correct answer is "sometimes".

Assertion example 1

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

    // Point A

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

        if (x != y) {
            // Point C
            z = z * 2;
        }

        // Point D

    }

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

For each assertion state if it is ALWAYS, NEVER, or SOMETIMES true at the specified points in the code.

.

	x < y	x == y	z == 0
Point A			
Point B			
Point C			
Point D			
Point E			

Assertion example 1

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

    // Point A

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

        if (x != y) {
            // Point C
            z = z * 2;
        }

        // Point D

    }

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

For each assertion state if it is ALWAYS, NEVER, or SOMETIMES true at the specified points in the code.

.

	x < y	x == y	z == 0
Point A	SOMETIMES	SOMETIMES	ALWAYS
Point B	NEVER	SOMETIMES	SOMETIMES
Point C	SOMETIMES	NEVER	NEVER
Point D	SOMETIMES	SOMETIMES	NEVER
Point E	ALWAYS	NEVER	SOMETIMES

Assertion example 2

```
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;
}
```

For each assertion state if it is ALWAYS, NEVER, or SOMETIMES true at the specified points in the code.

	next == 0	prev == 0	next == prev
Point A			
Point B			
Point C			
Point D			
Point E			

Assertion example 2

```
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;
}
```

For each assertion state if it is ALWAYS, NEVER, or SOMETIMES true at the specified points in the code.

	next == 0	prev == 0	next == prev
Point A	SOMETIMES	ALWAYS	SOMETIMES
Point B	NEVER	SOMETIMES	SOMETIMES
Point C	NEVER	NEVER	ALWAYS
Point D	SOMETIMES	NEVER	SOMETIMES
Point E	ALWAYS	SOMETIMES	SOMETIMES

Assertion example 3

```
public static int pow(int x, int y) {
    int prod = 1;
    if (y >= 0) {

        // Point A
        while (y > 0) {
            // Point B
            if (y % 2 == 0) {
                // Point C
                x = x * x;
                y = y / 2;
                // Point D
            } else {
                // Point E
                prod = prod * x;
                y--;
                // Point F
            }
        }
        // Point G
    }
    return prod;
}
```

For each assertion state if it is ALWAYS, NEVER, or SOMETIMES true at the specified points in the code.

	y > 0	y % 2 == 0
Point A		
Point B		
Point C		
Point D		
Point E		
Point F		
Point G		

Assertion example 3

```
public static int pow(int x, int y) {
    int prod = 1;
    if (y >= 0) {

        // Point A
        while (y > 0) {
            // Point B
            if (y % 2 == 0) {
                // Point C
                x = x * x;
                y = y / 2;
                // Point D
            } else {
                // Point E
                prod = prod * x;
                y--;
                // Point F
            }
        }
        // Point G
    }
    return prod;
}
```

For each assertion state if it is ALWAYS, NEVER, or SOMETIMES true at the specified points in the code.

	y > 0	y % 2 == 0
Point A	SOMETIMES	SOMETIMES
Point B	ALWAYS	SOMETIMES
Point C	ALWAYS	ALWAYS
Point D	ALWAYS	SOMETIMES
Point E	ALWAYS	NEVER
Point F	SOMETIMES	ALWAYS
Point G	NEVER	ALWAYS