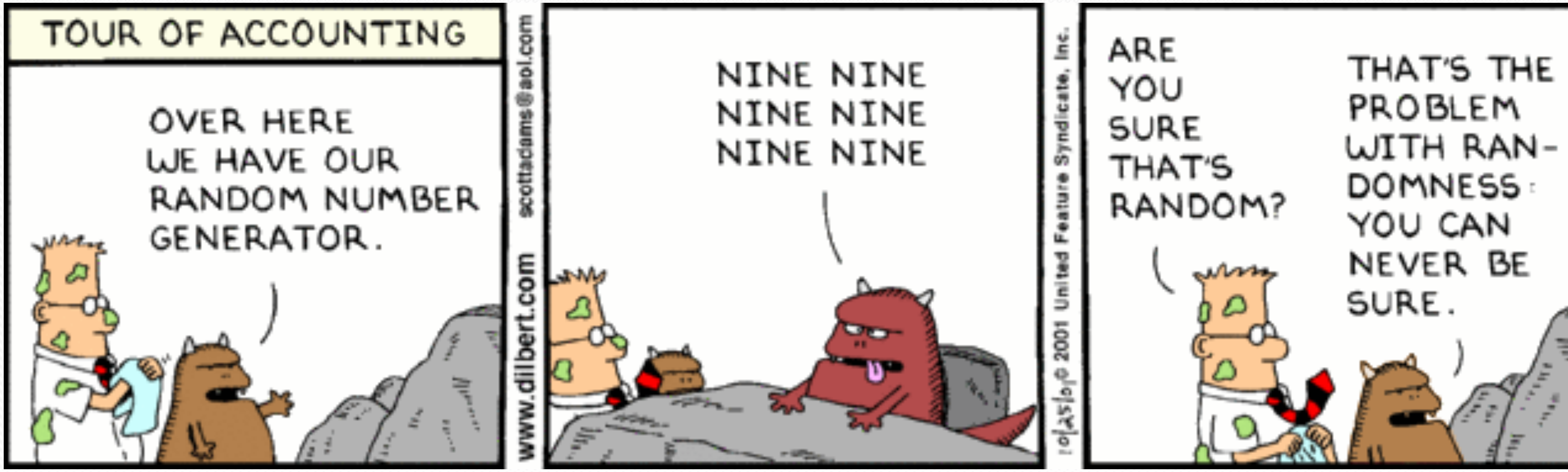
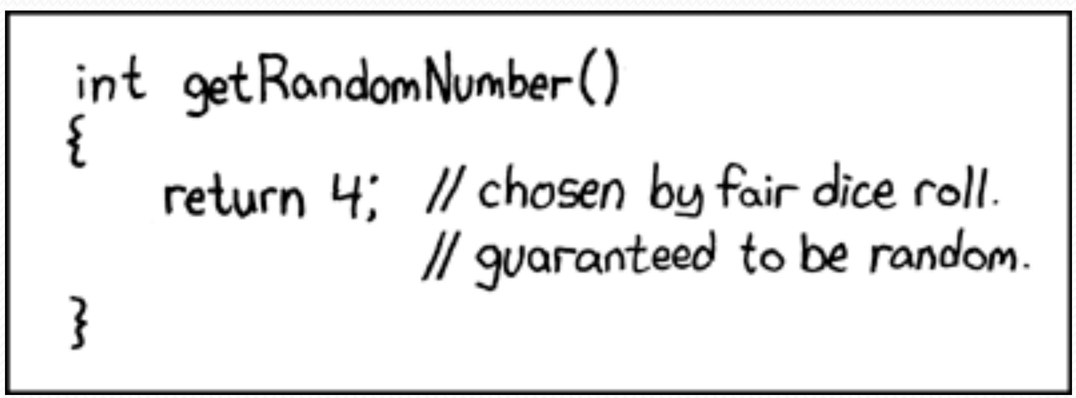
Building Java Programs

Chapter 5
Lecture 5-2: Random Numbers

reading: 5.1, 5.6





http://xkcd.com/221/

Randomness

- Lack of predictability: don't know what's coming next
- Random process: outcomes do not follow a deterministic pattern (math, statistics, probability)
- Lack of bias or correlation (statistics)
- Relevant in lots of fields
  - Genetic mutations (biology)
  - Quantum processes (physics)
  - Random walk hypothesis (finance)
  - Cryptography (computer science)
  - Game theory (mathematics)
  - Determinism (religion)

Pseudo-Randomness

- Computers generate numbers in a predictable way using a mathematical formula
- Parameters may include current time, mouse position
  - In practice, hard to predict or replicate
- True randomness uses natural processes
  - Atmospheric noise (http://www.random.org/)
  - Lava lamps (patent #5732138)
  - Radioactive decay

The `Random` class

- A `Random` object generates pseudo-random numbers.
  - Class `Random` is found in the `java.util` package.

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

Method name	Description
`nextInt()`	returns a random integer
`nextInt(`***max***`)`	returns a random integer in the range [0, *max*) in other words, 0 to *max*-1 inclusive
`nextDouble()`	returns a random real number in the range [0.0, 1.0)

  - Example:

    ```
    Random rand = new Random();
    int randomNumber = rand.nextInt(10);    // 0-9
    ```

Generating random numbers

- Common usage: to get a random number from 1 to *N*

```
int n = rand.nextInt(20) + 1;    // 1-20 inclusive
```

- To get a number in arbitrary range [*min*, *max*] inclusive:

  name`.nextInt(`***size of range***`) + `***min***

  - Where ***size of range*** is (***max*** `-` ***min*** `+ 1`)

  - Example: A random integer between 4 and 10 inclusive:

```
int n = rand.nextInt(7) + 4;
```

`Random` questions

- Given the following declaration, how would you get:

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

  - A random number between 1 and 47 inclusive?

```
int random1 = rand.nextInt(47) + 1;
```

  - A random number between 23 and 30 inclusive?

```
int random2 = rand.nextInt(8) + 23;
```

  - A random even number between 4 and 12 inclusive?

```
int random3 = rand.nextInt(5) * 2 + 4;
```

`Random` and other types

- `nextDouble` method returns a `double` between 0.0 - 1.0
  - Example: Get a random GPA value between 1.5 and 4.0:

```
double randomGpa = rand.nextDouble() * 2.5 + 1.5;
```

- Any set of possible values can be mapped to integers
  - code to randomly play Rock-Paper-Scissors:

```
int r = rand.nextInt(3);
if (r == 0) {
    System.out.println("Rock");
} else if (r == 1) {
    System.out.println("Paper");
} else {  // r == 2
    System.out.println("Scissors");
}
```

Random question

- Write a program that simulates rolling two 6-sided dice until their combined result comes up as 7.

```
2 + 4 = 6
3 + 5 = 8
5 + 6 = 11
1 + 1 = 2
4 + 3 = 7
You won after 5 tries!
```

Random answer

```
// Rolls two dice until a sum of 7 is reached.
import java.util.*;

public class Dice {
    public static void main(String[] args) {
        Random rand = new Random();
        int tries = 0;

        int sum = 0;
        while (sum != 7) {
            // roll the dice once
            int roll1 = rand.nextInt(6) + 1;
            int roll2 = rand.nextInt(6) + 1;
            sum = roll1 + roll2;
            System.out.println(roll1 + " + " + roll2 + " = " + sum);
            tries++;
        }

        System.out.println("You won after " + tries + " tries!");
    }
}
```

`Random` question

- Write a program that plays an adding game.
  - Ask user to solve random adding problems with 2-5 numbers.
  - The user gets 1 point for a correct answer, 0 for incorrect.
  - The program stops after 3 incorrect answers.

```
4 + 10 + 3 + 10 = 27
9 + 2 = 11
8 + 6 + 7 + 9 = 25
Wrong! The answer was 30
5 + 9 = 13
Wrong! The answer was 14
4 + 9 + 9 = 22
3 + 1 + 7 + 2 = 13
4 + 2 + 10 + 9 + 7 = 42
Wrong! The answer was 32
You earned 4 total points
```

Random answer

```
// Asks the user to do adding problems and scores them.
import java.util.*;

public class AddingGame {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        Random rand = new Random();

        // play until user gets 3 wrong
        int points = 0;
        int wrong = 0;
        while (wrong < 3) {
            int result = play(console, rand);   // play one game
            if (result == 0) {
                wrong++;
            } else {
                points++;
            }
        }

        System.out.println("You earned " + points + " total points.");
    }
```

Random answer 2

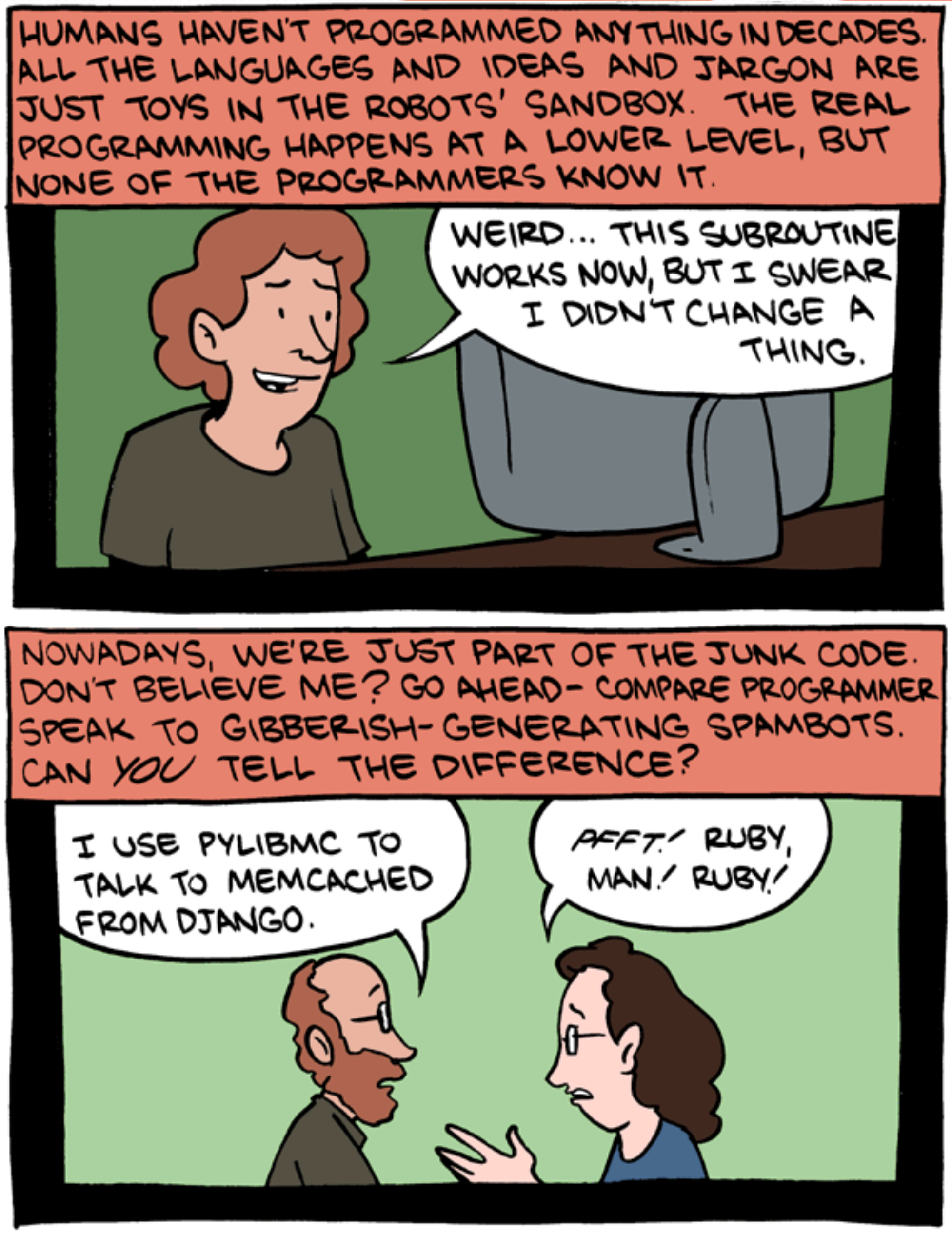
```
    ...
    // Builds one addition problem and presents it to the user.
    // Returns 1 point if you get it right, 0 if wrong.
    public static int play(Scanner console, Random rand) {
        // print the operands being added, and sum them
        int operands = rand.nextInt(4) + 2;
        int sum = rand.nextInt(10) + 1;
        System.out.print(sum);

        for (int i = 2; i <= operands; i++) {
            int n = rand.nextInt(10) + 1;
            sum += n;
            System.out.print(" + " + n);
        }
        System.out.print(" = ");

        // read user's guess and report whether it was correct
        int guess = console.nextInt();
        if (guess == sum) {
            return 1;
        } else {
            System.out.println("Wrong! The answer was " + total);
            return 0;
        }
    }
}
```

Building Java Programs

Chapter 5

Lecture 5-4: Assertions

reading: 5.5



Punchline to a longer comic: http://www.smbc-comics.com/index.php?db=comics&id=2362#comic

Logical assertions

- **assertion**: A statement that is either true or false.

  Examples:
  - Java was created in 1995.
  - The sky is purple.
  - 23 is a prime number.
  - 10 is greater than 20.
  - x divided by 2 equals 7. *(depends on the value of x)*

- An assertion might be false ("The sky is purple" above), but it is still an assertion because it is a true/false statement.

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 assertions

- 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
```

- 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
  - a parameter's initial value to 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".
  - If you're unsure, "Sometimes" is a good guess.

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);
}
```

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`
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;
}
```

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`
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

```
// Assumes y >= 0, and returns x^y
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 * x;
            y = y / 2;
            // Point D
        } else {
            // Point E
            prod = prod * x;
            y--;
            // Point F
        }
    }
    // Point G
    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`
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