int getRandomNumber()
{
  return 4;  // chosen by fair dice roll.
  // guaranteed to be random.
}

http://xkcd.com/221/
Chaining

- Good method decomposition
  ```java
  public static void main(){
    makeDough();
    bakeCookies();
    decorateCookies();
  }
  
  public static void makeDough(){
    retrieveIngredients();
    mixIngredients();
  }
  
  public static void bakeCookies(){
    placeOnCookieSheet();
    putInOven();
    wait10Minutes();
    pullOutCookies();
  }
  
  public static void decorateCookies(){
    ...
  }
  ```

- Chaining 1
  ```java
  public static void main(){
    makeDough();
    decorateCookies();
  }
  
  public static void makeDough(){
    retrieveIngredients();
    mixIngredients();
    bakeCookies();
  }
  
  public static void bakeCookies(){
    placeOnCookieSheet();
    putInOven();
    wait10Minutes();
    pullOutCookies();
  }
  
  public static void decorateCookies(){
    ...
  }
  ```

- Main is not a good summary
- “makeDough” does not describe the method well
Chaining

• Good method decomposition

```java
public static void main(){
    makeDough();
    bakeCookies();
    decorateCookies();
}
```

```java
public static void makeDough(){
    retrieveIngredients();
    mixIngredients();
}
```

```java
public static void bakeCookies(){
    placeOnCookieSheet();
    putInOven();
    wait10Minutes();
    pullOutCookies();
}
```

```java
Public static void decorateCookeis(){
    ...
}
```

• Chaining 2

```java
public static void main(){
    makeDoughAndBakeIt();
    decorateCookies();
}
```

```java
public static void makeDoughAndBakeIt(){
    retrieveIngredients();
    mixIngredients();
    bakeCookies();
}
```

```java
public static void bakeCookies(){
    placeOnCookieSheet();
    putInOven();
    wait10Minutes();
    pullOutCookies();
}
```

```java
Public static void decorateCookeis(){
    ...
}
```

- Control does not return to main when it should
- “makeDoughAndBakeIt” is unnatural task division
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.
  ```java
  import java.util.*;
  ```

<table>
<thead>
<tr>
<th>Method name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nextInt()</code></td>
<td>returns a random integer</td>
</tr>
<tr>
<td><code>nextInt(max)</code></td>
<td>returns a random integer in the range ([0, max))</td>
</tr>
<tr>
<td></td>
<td>in other words, 0 to (max-1) inclusive</td>
</tr>
<tr>
<td><code>nextDouble()</code></td>
<td>returns a random real number in the range ([0.0, 1.0))</td>
</tr>
</tbody>
</table>

- Example:

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

- To get a number in arbitrary range \([\text{min}, \text{max}]\) inclusive:
  ```java
  name.nextInt(size of range) + min
  ```
  Where \(\text{size of range}\) is \((\text{max} - \text{min} + 1)\)

- Example: A random integer between 4 and 10 inclusive:
  ```java
  int n = rand.nextInt(7) + 4;
  ```
Random questions

- Given the following declaration, how would you get:
  ```java
  Random rand = new Random();
  ```

  - A random number between 1 and 47 inclusive?
    ```java
    int random1 = rand.nextInt(47) + 1;
    ```

  - A random number between 23 and 30 inclusive?
    ```java
    int random2 = rand.nextInt(8) + 23;
    ```

  - A random even number between 4 and 12 inclusive?
    ```java
    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:
    ```java
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:
    ```java
    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.

\[
\begin{align*}
2 + 4 &= 6 \\
3 + 5 &= 8 \\
5 + 6 &= 11 \\
1 + 1 &= 2 \\
4 + 3 &= 7 \\
\text{You won after 5 tries!}
\end{align*}
\]
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
// 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;  
   }  
}  
}