"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 of 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
The keyword list thus far:

- Complete list of Java keywords:
  
<table>
<thead>
<tr>
<th>Keyword</th>
<th>Keyword</th>
<th>Keyword</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>abstract</td>
<td>default</td>
<td>if</td>
<td>private</td>
</tr>
<tr>
<td>boolean</td>
<td>do</td>
<td>implements</td>
<td>protected</td>
</tr>
<tr>
<td>break</td>
<td>double</td>
<td>import</td>
<td>public</td>
</tr>
<tr>
<td>byte</td>
<td>else</td>
<td>instanceof</td>
<td>return</td>
</tr>
<tr>
<td>case</td>
<td>extends</td>
<td>int</td>
<td>short</td>
</tr>
<tr>
<td>catch</td>
<td>final</td>
<td>interface</td>
<td>static</td>
</tr>
<tr>
<td>char</td>
<td>finally</td>
<td>long</td>
<td>strictfp</td>
</tr>
<tr>
<td>class</td>
<td>float</td>
<td>native</td>
<td>volatile</td>
</tr>
<tr>
<td>const</td>
<td>for</td>
<td>new</td>
<td>switch</td>
</tr>
<tr>
<td>continue</td>
<td>goto</td>
<td>package</td>
<td>synchronized</td>
</tr>
<tr>
<td>assert</td>
<td>enum</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Methods that are tests

- Some methods return logical values (true or false).
  - A call to such a method is used as a `<test>` in a loop or if.

    ```java
    Scanner console = new Scanner(System.in);
    System.out.print("Type your first name: ");
    String name = console.next();

    if (name.startsWith("Dr.")) {
        System.out.println("Med school or PhD?");
    } else if (name.endsWith("Esq.")) {
        System.out.println("And I am Ted 'Theodore' Logan");
    }
    ```

Strings question

- Prompt the user for two words and report whether they:
  - "rhyme" (end with the same last two letters)
  - alliterate (begin with the same letter)

    - Example output: (run #1)
      Type two words: `car STAR`
      They rhyme!

    - (run #2)
      Type two words: `bare bear`
      They alliterate!

    - (run #3)
      Type two words: `sell shell`
      They alliterate!
      They rhyme!

    - (run #4)
      Type two words: `extra strawberry`
Strings answer

// Determines whether two words rhyme and/or alliterate.
import java.util.*;
public class Rhyme {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.print("Type two words: ");
        String word1 = console.next().toLowerCase();
        String word2 = console.next().toLowerCase();
        // check whether they end with the same two letters
        if (word2.length() >= 2 &&
                word1.endsWith(word2.substring(word2.length() - 2))) {
            System.out.println("They rhyme!");
        }
        // check whether they alliterate
        if (word1.startsWith(word2.substring(0, 1))) {
            System.out.println("They alliterate!");
        }
    }
}

Random numbers

Example:
Generating random numbers

- Common usage: to get a random number from 1 to $N$
  \[
  \text{int } n = \text{rand.nextInt}(20) + 1; \\
  \text{// 1-20 inclusive}
  \]

- To get a number in arbitrary range $[\text{min}, \text{max}]$ inclusive:
  \[
  \text{<name>.nextInt(<size of range>) + <min>}
  \]
  - Where $\text{size of range}$ is $(\text{max} - \text{min} + 1)$
  - Example: A random integer between 4 and 10 inclusive:
    \[
    \text{int } n = \text{rand.nextInt}(7) + 4;
    \]

The Random class

- A Random object generates pseudo-random numbers.
  - Class Random is found in the java.util package.
    \[
    \text{import java.util.Random;}
    \]

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

- Example:
  \[
  \text{Random rand = new Random();}
  \text{int randomNumber = rand.nextInt(10);} // 0-9
  \]
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)
  
  ```java
  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
  ```java
  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 of two 6-sided dice until their combined result comes up as 7.
  ```java
  // 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 answer

- You won after 5 tries!
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.

```java
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.");
    }

    // 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) {
        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;
        }
    }
}
```