Solutions: for vs. while

Exercise 1: calculate the sum of 10 numbers

Not only the amount of numbers is known before the program reaches the loop, but it's known by the programmer when they write the code. We say it's known at *compile time*.

This is the perfect case for a `for`:

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);
        int sum = 0;

        // VERY GOOD: `i` stays inside the loop and is fully managed by the `for` statement
        for (int i = 1; i <= 10; i++) {
            System.out.print("Please enter number " + i + ": ");
            int x = input.nextInt();
            sum += x;
        }

        System.out.println("The sum of the numbers is: " + sum);
    }
}
```

It's also possible to solve it using `while`, but the result is not as elegant as using `for`:

```
import java.util.Scanner;
```

```
public class Main {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);
        int sum = 0;
        // BAD: `i` is defined outside the loop and we must use it in 3 different places
        int i = 1; // 1st

        // BAD: we should use `for` in this case
        while (i <= 10) { // 2nd
            System.out.print("Please enter number " + i + ": ");
            int x = input.nextInt();
            sum += x;
            i++; // 3rd
        }

        System.out.println("The sum of the numbers is: " + sum);
    }
}
```

Exercise 2: calculate the sum of N numbers

The amount of numbers is known because we ask the user before the program reaches the loop. It's not known by the programmer when they write the code, but it's known by the program.

This is still the perfect case for a `for`:

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);
        int sum = 0;

        System.out.print("How many numbers? ");
```

```
        int n = input.nextInt();

        // VERY GOOD: `i` stays inside the loop and is fully managed by the `for` statement
        for (int i = 1; i <= n; i++) {
            System.out.print("Please enter number " + i + ": ");
            int x = input.nextInt();
            sum += x;
        }

        System.out.println("The sum of the numbers is: " + sum);
    }
}
```

As before, it's also possible to solve it using `while`, but it's not as elegant as using `for`:

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);
        int sum = 0;
        // BAD: `i` is defined outside the loop and we must use it in 3 different places
        int i = 1; // 1st

        System.out.print("How many numbers? ");
        int n = input.nextInt();

        // BAD: we should use `for` in this case
        while (i <= n) { // 2nd
            System.out.print("Please enter number " + i + ": ");
            int x = input.nextInt();
            sum += x;
            i++; // 3rd
        }

        System.out.println("The sum of the numbers is: " + sum);
```

```
    }
}
```

Exercise 3: calculate the sum of some numbers

The amount of numbers is not known before the program reaches the loop, because the user can decide at any time to stop by entering zero. So we, and the program, cannot know how many numbers to process before entering the loop.

This is good case for a `while`:

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);
        int sum = 0;

        System.out.print("Please enter a number: ");
        int x = input.nextInt();

        // GOOD: we don't know how many iterations, we can't use `for`
        while (x != 0) {
            sum += x;
            System.out.print("Please enter a number: ");
            x = input.nextInt();
        }

        System.out.println("The sum of the numbers is: " + sum);
    }
}
```

As before, it's also possible to solve it using `for`, but it's not elegant as using `while` or `do` ... `while`:

```java
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);
        int sum = 0;

        System.out.print("Please enter a number: ");

        // UNUSUAL: we consider this use of `for` unusual and kind of for expert users:
        //          although it's 100% correct and works, we suggest using `while` instead
        for (int x = input.nextInt(); x != 0; x = input.nextInt()) {
            sum += x;
            System.out.print("Please enter a number: ");
        }

        System.out.println("The sum of the numbers is: " + sum);
    }
}
```

Look back at the implementation with `while`: do you see the sentence “Please enter a number” written twice? The `while` code above is good, but we can still work on it. This is a rare case where the code can be further improved using `do` … `while`:

```java
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);
        int sum = 0;
        int x;

        // GOOD: we don't know how many iterations, we can't use `for`.
```

```
        //       we can use `do`/`while` because we know we ask the user at least once
        do {
            System.out.print("Please enter a number: ");
            x = input.nextInt();
            sum += x;
        } while (x != 0);

        System.out.println("The sum of the numbers is: " + sum);
    }
}
```

However, the improvements `do` … `while` brings over `while` are minimal and this statement introduces new issues (see that `x` unassigned?). It's perfectly fine to solve this exercise using just `while`.

Number guessing game

We're building a game, the user has to guess a random number between 1 and 10 the computer thought of.

First, how do we get a random number, there are several ways. One easy way is to get a random number between 0.0 (inclusive) and 1.0 (exclusive) and multiply that number by 10 to get numbers from 0 (inclusive) to 9 (inclusive). Think a bit why is that: `0.0 * 10 = 0.0` converted to `int` is `0`, `0.99999 * 10 = 9.99999` (there could be more 9s after the dot) and `9.99999` converted to `int` is `9`. To get numbers from 1 to 10, we just add 1 to that random number we got.

Next we need to think about when the game ends, and that's when the user has no tries left or the user guessed five random numbers. So we need to loop until user is out of guesses or he has guessed five numbers. So those are two numbers we need to keep track of.

We also need to think about when are we generating the new random number for the user to guess, when are we increasing the number of guesses left vs when we are decreasing the number of guesses left.

```
import java.util.Scanner;

class Main {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
```

```
  int allowedNumberOfGuesses = 3; // at the start user has 3 guesses
  int correctGuesses = 0; // and no correct guesses
  int numberToGuess = randomNumberToGuess(); // this is the first number to try
  while (allowedNumberOfGuesses != 0 && correctGuesses != 5) { // if the user is out of guesses or has
    System.out.println("I've thought of a number, can you guess what number it is?");
    int guess = scanner.nextInt();

    if (guess == numberToGuess) {
      // here the user guessed the number, we need to increase the allowed number of guesses,
      // increase the counter of guesses and compute a new random number for the user to guess
      allowedNumberOfGuesses++;
      correctGuesses++;
      numberToGuess = randomNumberToGuess(); // compute new number to guess

      System.out.println("Correct! You get one more try, you have " + allowedNumberOfGuesses + " tries
    } else {
      // we decrease the number of allowed guesses,
      // BUT we keep the the same number for the user to try and guess it again
      allowedNumberOfGuesses--;
      System.out.println("Not the number I thought of! Number of attempts decreased, you have " + allow
    }
  }

  // we reach this point only when there are no more allowed attempts
  // or the user has guessed 5 numbers
  if (allowedNumberOfGuesses == 0) {
    System.out.println("You lost the game no more guesses left!");
  } else if (correctGuesses == 5) {
    System.out.println("You won the game!");
  }
}

public static int randomNumberToGuess() {
  // this will give us a random number from 1 to 10
  // Math.random() gives a random number from 0.0 to 1.0
  // remember 0.9 multiplied by 10 in int type equals 9
  return 1 + (int) (Math.random() * 10);
```

```
    }
}
```

Made with ❤ by teachers at ReDI School.