LAB2 - Java Basics

COMP-2021: Object-Oriented Programming

Objective

- To utilize Java basic elements to implement simple algorithms

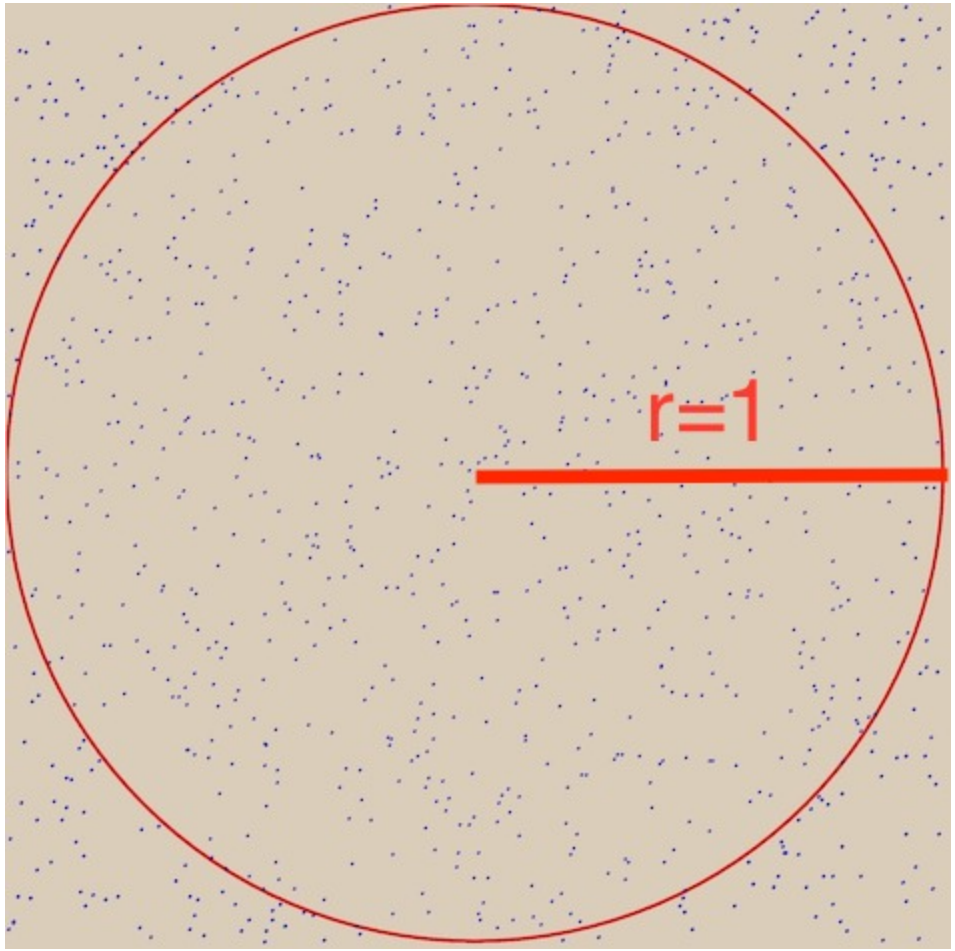
Examples

- Monte Carlo Simulation
- Find Prime Numbers

Practice

- Bubble Sort

Monte Carlo Simulation



The areas of the circle and the square are:

```
circle_area = π * 1 = π

square_area = 2 * 2 = 4
```

The `ratio` of the area of the circle to the area of the square is

```
ratio = circle_area / square_area = π / 4
```

The ratio of the number of points that fall inside the circle to the total number of points is equal to the ratio of the two areas.

```
π = 4 * ratio
π = 4 * circle_area/square_area
```

The accuracy of the ratio depends on the number of generated points, with more points leading to a more accurate value.

Simulation of π

The `ratio` is equal to the probability `p` of a point within the square being also inside the circle.

```
ratio = p
```

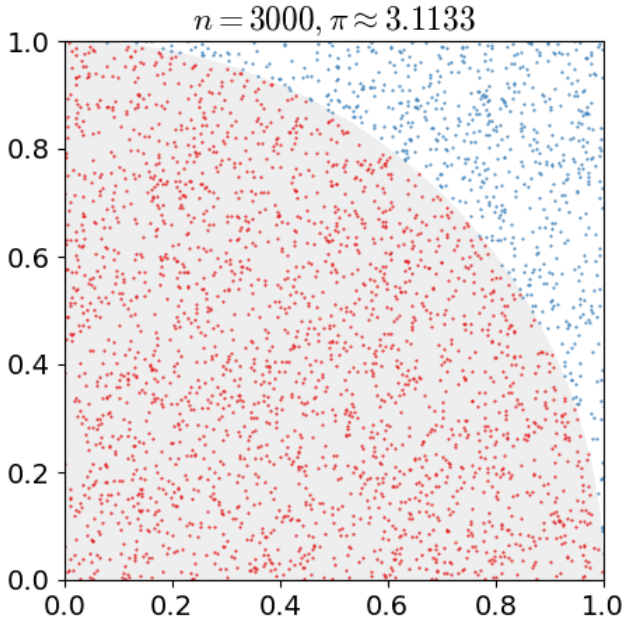
We randomly pick `n` locations in the square, let `hitCount` be the number of these locations that are inside the circle.
The larger the total number of locations picked, the closer `hitCount / n` is to `p`.

```
p ≈ hitCount / n
```

```
π / 4 ≈ hitCount / n

π ≈ 4 * hitCount / n
```

Simulation of π



- The probability of a point located in the circle is the ratio of the two areas.
- If we divide the figure in to 4 parts equally, the ratio and probability remains the same.The more random points generated, the more accuracy we get.

What to implement?

1. Generate a random point (x,y) within the square range [0,1];
2. Calculate the distance between (x,y) and center (0,0);
3. Update the counting variable `hitCount` if the distance is smaller than the radius `r=1` ;
4. Repeat 1, 2 and 3 for `n` times ;
5. Calculate the approximate value of probability `hitCount / n` ;
6. Estimate π value `pi` .

Hints

- Generate a random double numbers between 0 and 1:

```
Random random = new Random();
// `import java.util.Random;` is required
double x = random.nextDouble();
```

- Calculate the distance between two points:

```
double distence = Math.sqrt(Math.pow(x, 2) + Math.pow(y, 2));
```

- Estimate π value `pi`:

```
double pi = 4.0 * hitCount / n;
```

1. to represent a point (x,y) we need two random numbers.
2. to Determine if a point is located within the circle area, we
    - calculate the distance between the point and the center;
    - check wether the distance is smaller than the radius `r=1`
3. Update the counting variable `hitCount` if the distance is smaller than 1;
4. Repeat 1, 2 and 3 for `n` times
5. Calculate the approximate value of probability `hitCount / n`.
6. Estimate π value `pi`.

Print Prime Numbers

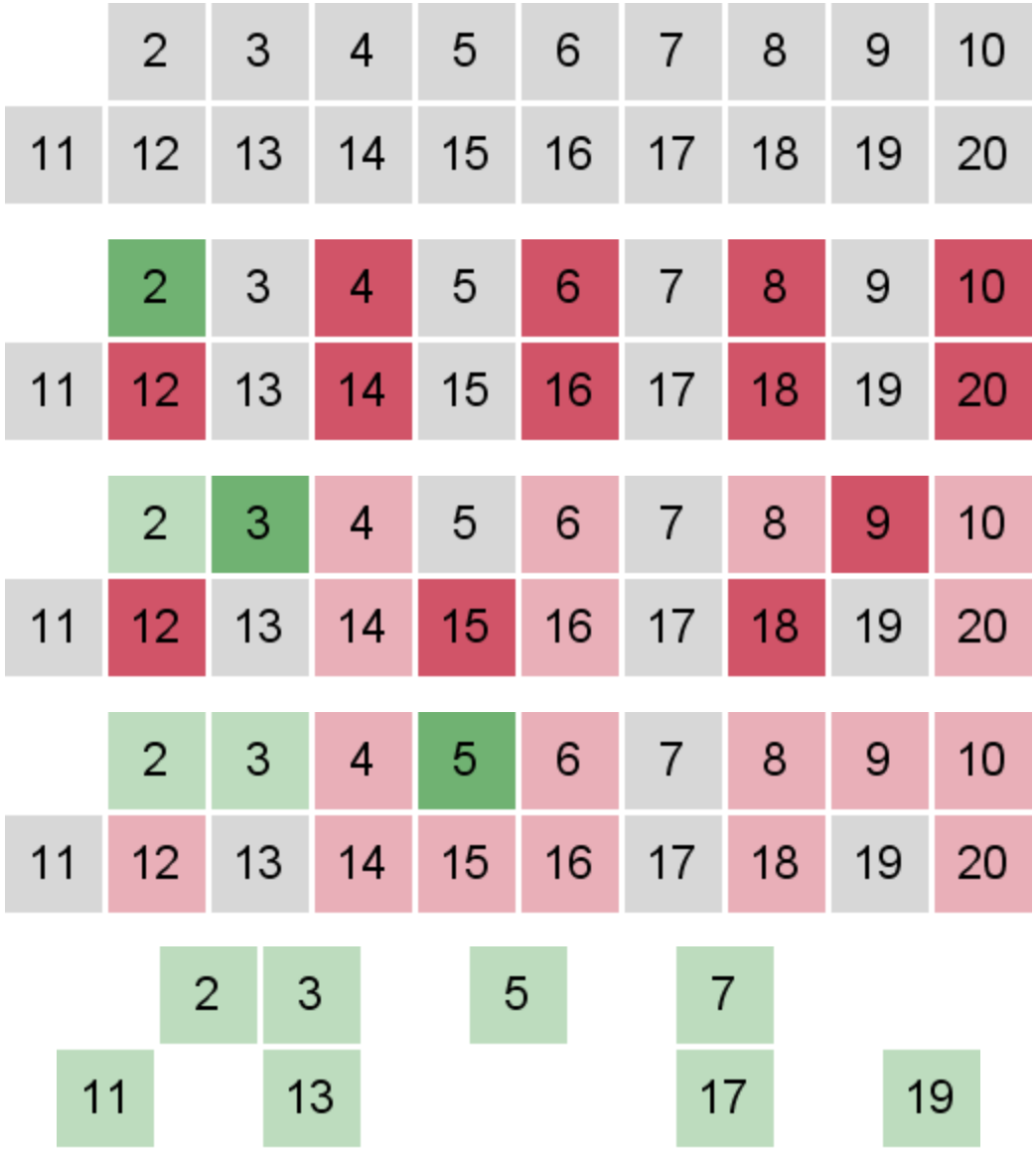
Write a program to find prime numbers that are less than or equal to a given integer `n`.

Prime Number

A prime number is a natural number which is greater than 1 and has no positive divisors other than 1 and itself.

Mark Composite Numbers

Given all positive integers within `n`, we iteratively mark the multiples of all prime numbers as composite (i.e., not prime), starting with the first prime number 2.



- When the algorithm terminates, all the unmarked numbers in the array are primes below `n`.

Summary and Hints

1. Create an array of consecutive integers from 2 through `n`: (2, 3, 4, ..., n).
2. Initially, let `p` be equal to 2, the smallest prime number.
3. Enumerate the multiples of `p` by counting from `2p` to `n` in step of `p`

   (e.g. `2p`, `3p`, `4p`, ...), and mark them in the list;

   `p` itself is the next prime and should not be marked.
4. Find the first unmarked number `unmarked` greater than `p` in the list.
5. Let `p` be equal to `unmarked` and repeat step 3, 4 and 5.
   If there is no such `unmarked`, the algorithm terminates.

We need two loops that

- one keep finding "next prime"
- the other enumerating the multiples of a prime and marking the list.

Hints

- A boolean array `new boolean[n+1]` is the simplest data structure that can be used to store a list of markable consecutive integers.
- Two loops:
  * One loop to find the next prime.
  * Another loop to mark the multiples of the prime.
- After marking, a prime number keeps initial value `false` while a composite number is marked as `true`.

reading commandline input

- Read one line of input from the command line.

```
public static String readUserInput(String msg) {
        System.out.println(msg);
        Scanner scanIn = new Scanner(System.in);
        // `import java.util.Scanner;` is required
        String inputString = scanIn.nextLine();
        scanIn.close();

        System.out.println("Enter String: " + inputString);
        return inputString;
    }
```

  - Usage:

```
String userInput = readUserInput("Input a number:");
```

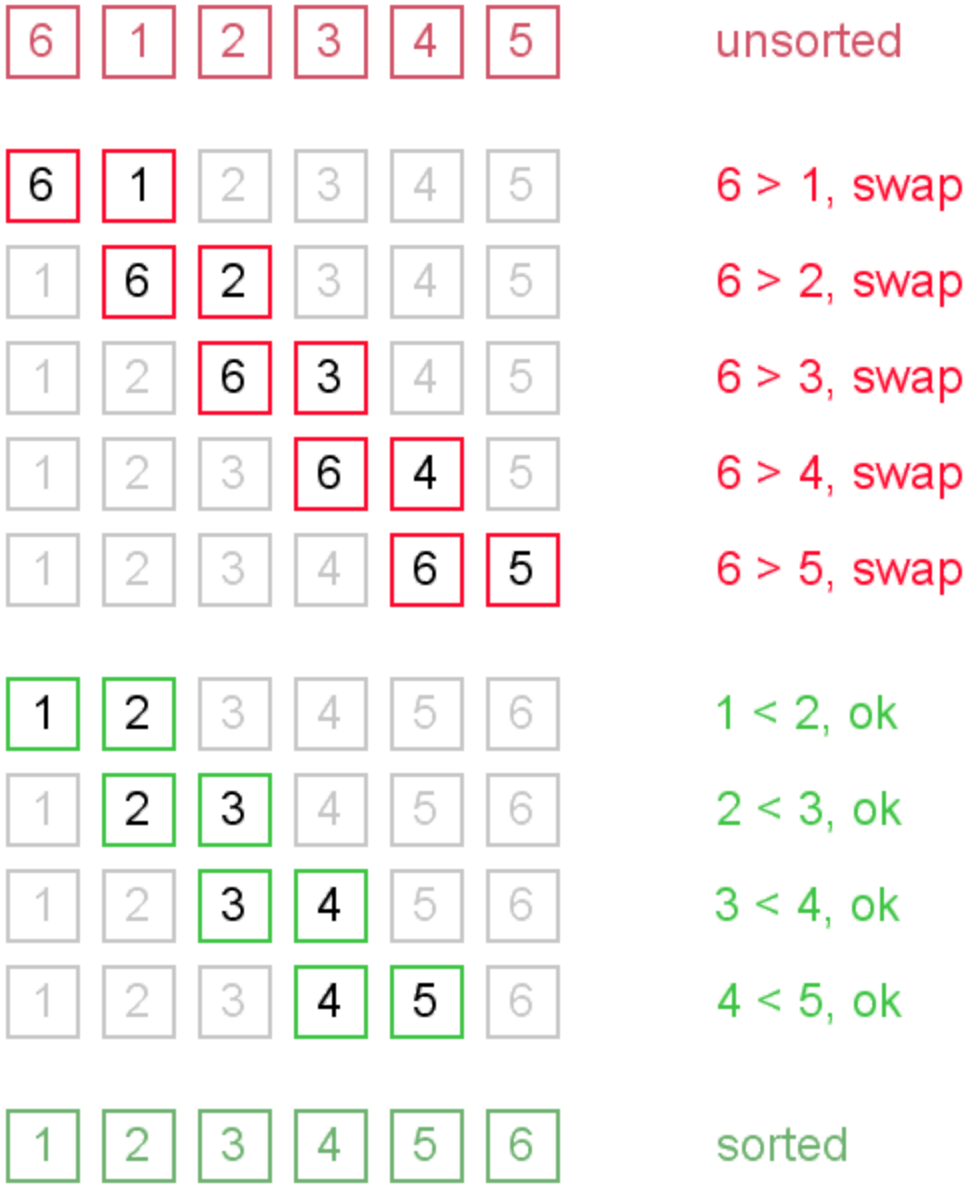
Practice - Bubble Sort

- Bubble sort compares each pair of adjacent items and swaps them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted.

1. Compare each pair of adjacent elements from the beginning of an array and, if they are in reversed order, swap them.
2. If at least one swap has been done, repeat from step 1.

- You can imagine that every iteration makes a big bubble float to the surface and stay there. The algorithm terminates when no bubble moves. The picture in the following page shows the process of sorting an array.

Practice - Bubble Sort



References

- https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes
- http://www.algolist.net/Algorithms/Number_theoretic/Sieve_of_Eratosthenes
- http://introcs.cs.princeton.edu/java/98simulation/
- https://en.wikipedia.org/wiki/Monte_Carlo_method#/media/File:Pi_30K.gif