GUI によるランダムウォークシミュレーション

7 月 5 日

3 年 14 組 4 番　新井健太

１．プログラム概要

0 ＜ p ＜ 1 を満たす実数 p があり、ある変数 x を確率 p で+1、確率 1-p で-1 する試行を繰り返し行うことを通称”ランダムウォーク”という。今回はこの確率 p と試行回数を自由に変更し、それによってランダムウォークの結果がどのように変化するかを視覚的にとらえることができる Java GUI プログラムを作成した。

２．開発環境/実行環境

マシン：PC

OS：Windows 10

言語：Java (Eclipse IDE version 2021-03 (4.19.0))

３．ファイル構成

Random.java

RandomWalkFrame.java

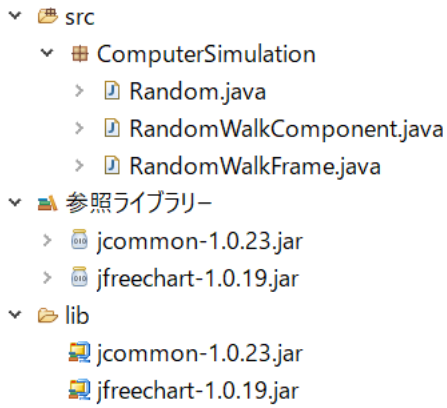
RandomWalkComponent.java

Jcommon-1.0.23.jar

Jfreechart-1.0.19.jar

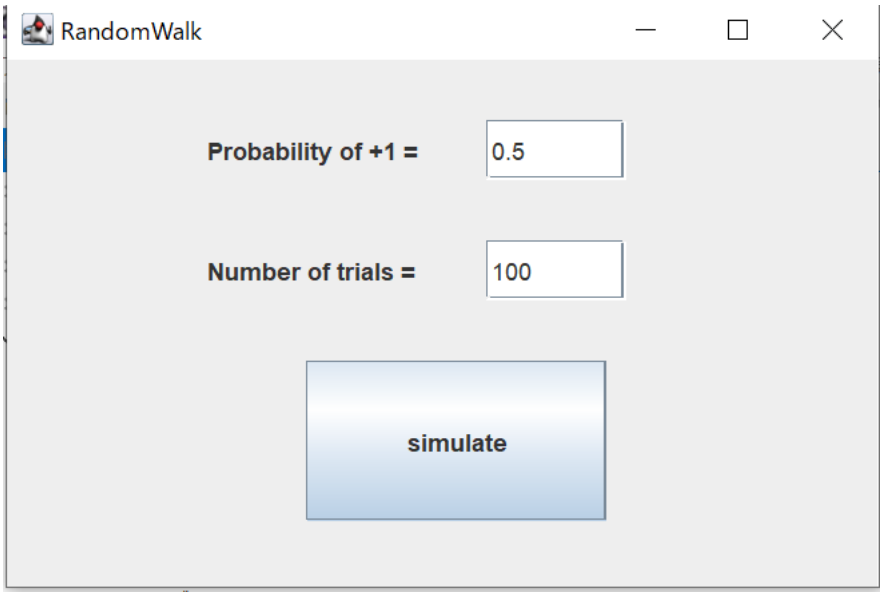
４．コンパイル方法

５．実行方法

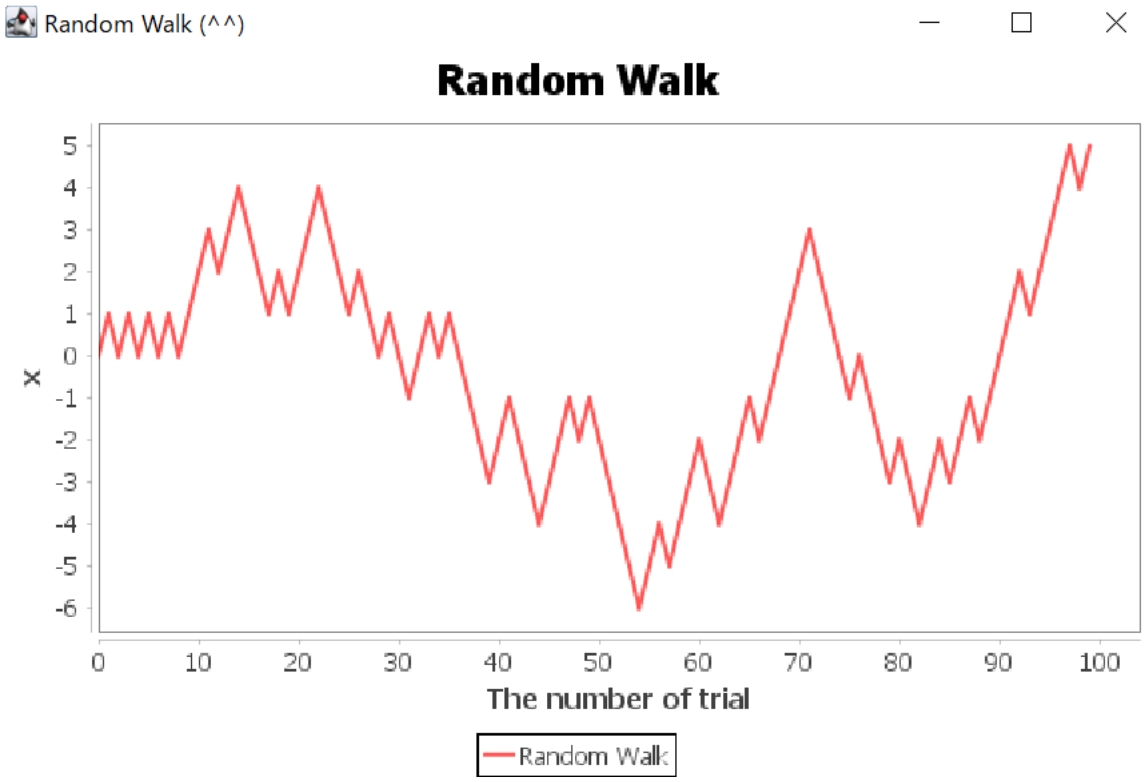


Eclipse で図のようにファイルを配置し、RandomWalkFrame をアプリケーションで実行する。Lib にインストールした jfreechart に関する二つのファイルを置き、ビルドパスに追加することで、二つのファイルを参照できるようになる。

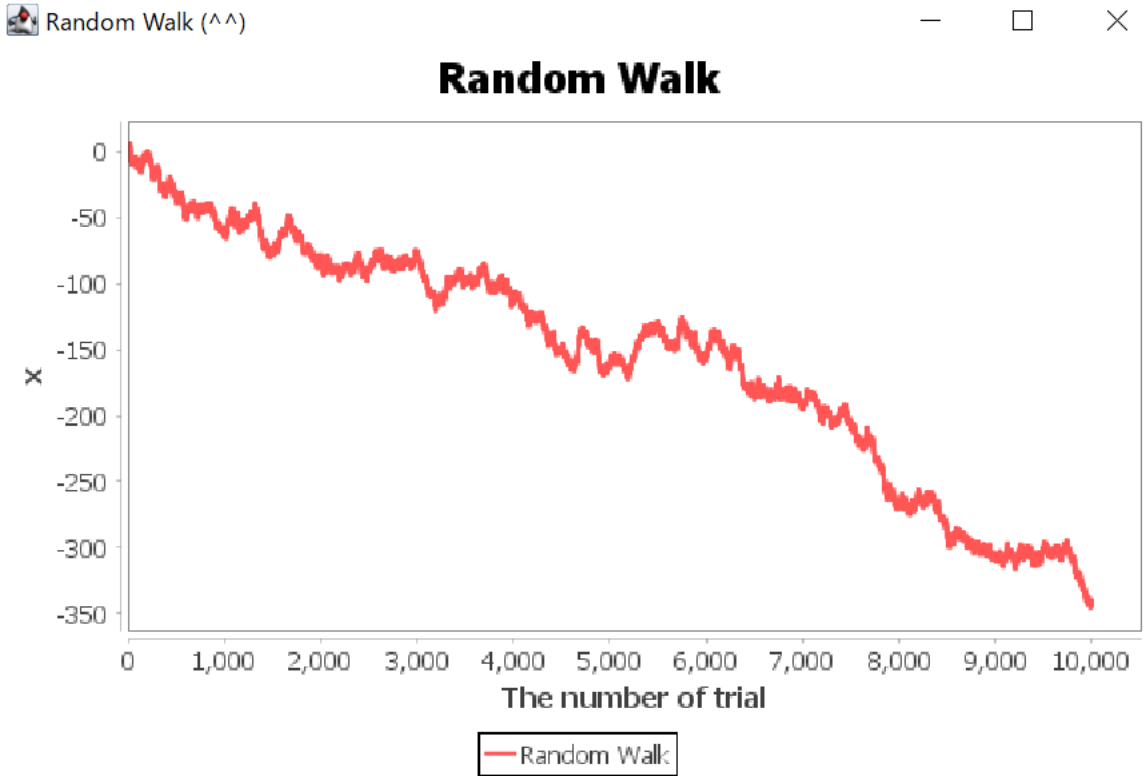
６．実行結果



↑RandomWalkFrame を eclipse 上で実行したときに表示される画面



↑ 0.5, 100 で simulate した結果



↑ 0.49, 10000 で simulate した結果

７．プログラム操作説明

起動した GUI 画面で Probability of +1 = の欄にランダムウォークの上昇確率を入力する。
Number of trials = にはランダムウォークの試行回数を入力する。

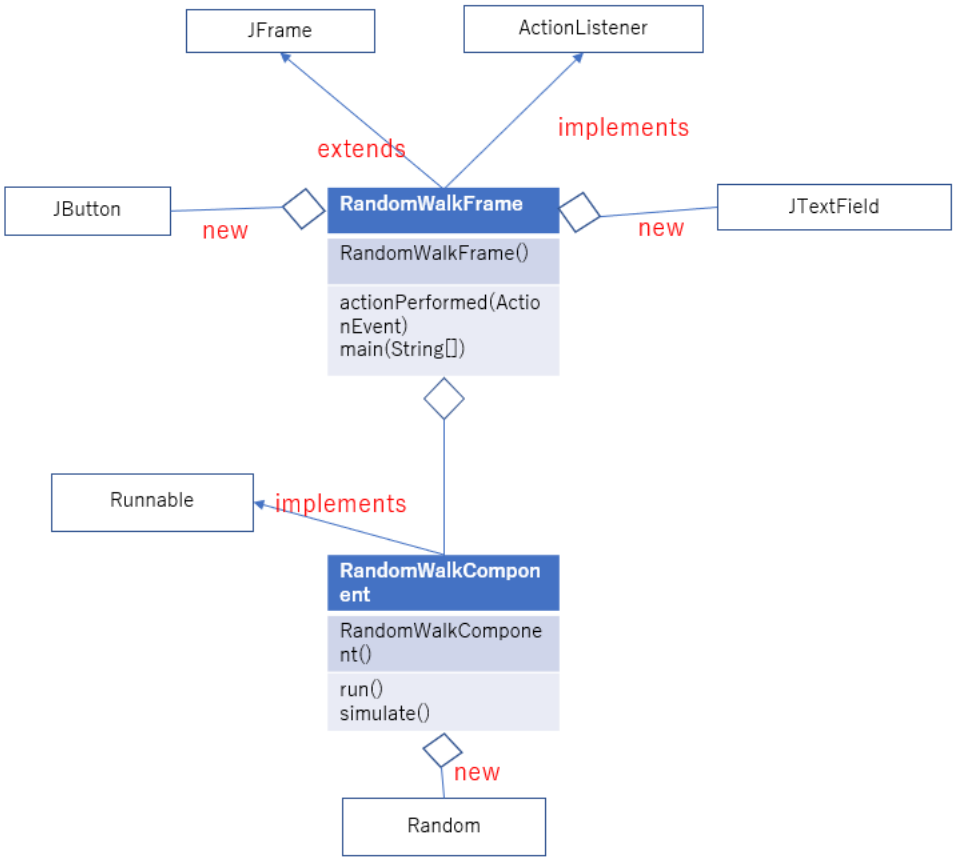
Simulate ボタンを押すと、入力した情報に基づいてランダムウォークの結果がグラフで表示される。

８．原理

Math.random()で 0 から 1 の乱数を生成し、確率 p との値を比較して+1 もしくは-1 を試行回数分だけ配列に格納する。その配列をもとにグラフを生成する。

９．プログラムコード説明

９．１：クラス図



以下にソースコードを添付する。

９．２クラス説明

９．２．１：Class RandomWalkFrame

```java
package ComputerSimulation;
import java.awt.Container;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
```

Import ファイルが多いのでここに掲載する。

↓RandomWalkFrame メソッドの概要

```java
public class RandomWalkFrame extends JFrame implements ActionListener {
    private JTextField pField = new JTextField("0.5");
    private JTextField wField = new JTextField("100");
    private JButton simBtn = new JButton("simulate");
    private RandomWalkComponent rwc = new RandomWalkComponent();

    public RandomWalkFrame() {
        super("RandomWalk");
        Container cont = getContentPane();
        cont.setLayout(null);

        JLabel pLabel = new JLabel("Probability of +1 = ");
        pLabel.setLocation(100, 30);
        pLabel.setSize(130, 30);
        cont.add(pLabel);

        pField.setLocation(240, 30);
        pField.setSize(70, 30);
        cont.add(pField);

        JLabel wLabel = new JLabel("Number of trials = ");
        wLabel.setLocation(100, 90);
        wLabel.setSize(130, 30);
        cont.add(wLabel);

        wField.setLocation(240, 90);
        wField.setSize(70, 30);
        cont.add(wField);

        simBtn.setLocation(150, 150);
        simBtn.setSize(150, 80);
        cont.add(simBtn);
        simBtn.addActionListener(this);

        setSize(450, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    }
```

ここでは GUI 起動時のフレームワークを定義する。各ラベルやボタン、テキストフィールドは Container に追加し、GUI のフレームに表示されるようにする。

```
public void actionPerformed(ActionEvent ae) {
    Object obj = ae.getSource();

    if(obj == simBtn) {
        try {
            int trialNum = Integer.parseInt(wField.getText());
            double prob = Double.parseDouble(pField.getText());
            rwc.simulate(trialNum, prob);
        }
        catch(NumberFormatException nfe) {}
    }

}

public static void main(String[] args) {
    // TODO 自動生成されたメソッド・スタブ
    new RandomWalkFrame();
}
```

simBtn が押された時の動作を actionPerformed に定義する。Obj はフレームに入力された値を保持しており、それらを変数に格納し、RandomWalkComponent のインスタンスである rwc の simulate に渡し、シミュレーションを実行する。

９．２．２：Class RandomWalkComponent

```
package ComputerSimulation;

import java.awt.BasicStroke;
import java.awt.Color;

import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartFrame;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;

public class RandomWalkComponent implements Runnable {
    private Random rnd;
    private int trialNum;

    public void simulate(int trialNum, double prob) {
        this.trialNum = trialNum;
        rnd = new Random(trialNum, prob);
        Thread th = new Thread(this);
        th.start();
    }
```

新たにスレッドを作成するため、Runnable インターフェースを実装する。Runnable インターフェースは run()メソッドの実装が必要である。Simulate()メソッドは RandomWalkFrame から呼び出されるメソッドで、試行回数と確率をフレームから受け取り、スレッドを開始する。

Run メソッドでは受け取った値から実際にグラフを生成するプロセスを記述する。

```
public void run() {
    int[] walkTraj = rnd.getTraj();
    XYSeries xy = new XYSeries("Random Walk");
    int xValue = 0;
    for (int i = 0; i < trialNum; i++) {
        xy.add(i, xValue);
        xValue += walkTraj[i];
    }
    XYSeriesCollection xyc = new XYSeriesCollection();
    xyc.addSeries(xy);

    JFreeChart chart = ChartFactory.createXYLineChart("Random Walk", "The number of trial", "x", xyc,
    chart.getXYPlot().getRenderer().setSeriesStroke(0, new BasicStroke(2));
    chart.getPlot().setBackgroundPaint(Color.WHITE);


    ChartFrame frame = new ChartFrame("Random Walk (^^)", chart);
    frame.setVisible(true);
    frame.setSize(600, 400);
```

getTraj は各試行において+1 か、-1 かを定義した配列を返す。xValue が実際に変動する値で、for 文によって walkTraj の各要素を足しながらグラフに反映している。

９．２．３：Class Random

```
public class Random {
    private int trialNum = 100;

    private int[] walkTraj;
    private double prob;

    public Random(int trialNum, double prob) {
        this.trialNum = trialNum;
        this.prob = prob;
        compute();
    }

    public void compute() {
        walkTraj = new int[trialNum];
        for (int i = 0; i < trialNum; i++) {
            if(Math.random() < prob) {
                walkTraj[i] = 1;
            }
            else {
                walkTraj[i] = -1;
            }
        }
    }

    public int[] getTraj() {
        return walkTraj;
    }

}
```

Random()コンストラクタではインスタンスに入力値を渡したのち、compute メソッドを呼び出す。Compute()メソッドでは入力された確率に基づいて walkTraj 配列に+1 か-1 を代入する。すなわち Random クラスのインスタンスを生成した時点で walkTraj にランダムウォーク情報が格納される。WalkTraj は getTraj を用いることで得ることができる。

１０．考察

・JfreeChart を導入したことでグラフをよりビジュアルに表現した。Canvas 上でのグラフ表現はグラフの概形のみに興味があるときに便利だが、ランダムウォークは最終的な値も重要であるため、JfreeChart を用いた。

・Simulate ボタンを押すたびに新たなスレッドによってランダムウォークを実行しグラフが表示されるため、条件を変えた時にどのようにグラフが変化するかをウィンドウ同士の比較で簡単に行える。

１１．所感

自分の想定通りに GUI を組み立てられると感動する。Run メソッド中のグラフ描画に関する記述に関しては自分でも何かいているかがよくわからなかったので、ネット記事のコードを 9 割がた丸写ししたが、一発でグラフが表示された時は驚きと喜びが隠し切れなかった。

１２．参考文献

・ランダムウォークのシミュレーション

https://www.ishikawa-lab.com/montecarlo/6shou.html

・イベントモデルについて

https://www.javadrive.jp/tutorial/event/index1.html

・JFreeChart を使って Java で様々なグラフを簡単に描画する方法

https://rainbow-engine.com/introduction-jfreechart-helloworld/