Event

オブジェクト指向プログラミング特論

2016年度

只木進一：工学系研究科

Event駆動

- イベント
  - マウスの挙動、キーボード操作、周辺機器の状態変化、プログラム要素の状況の変化などなど
- イベントを契機に動作することをイベント駆動と言う

JavaとEvent

- GUIプログラム内の例
  - マウスの操作によって、特別なメソッドが起動される
- 要点
  - イベントの定義
  - イベントを受けて動作する側(listener)の登録
  - イベント発生時の動作の記述

guiWithAction.MainFrameクラス（復習）

- initComponentsメソッド内
  - ボタンにリスナーを登録
  - リスナクラスに注意

```
jButton1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
           jButton1ActionPerformed(evt);
        }
    });
```

- 具体的動作定義

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
    System.out.println(evt.getActionCommand()+"が押されました");
  }
```

構成要素：イベント

- java.util.EventObjectの派生クラス
  - java.util.EventObjectは、イベント発生元情報だけを保持
- 必要な情報を伝達するために拡張
  - イベントの種類
  - 発生した場所

構成要素：イベントリスナー

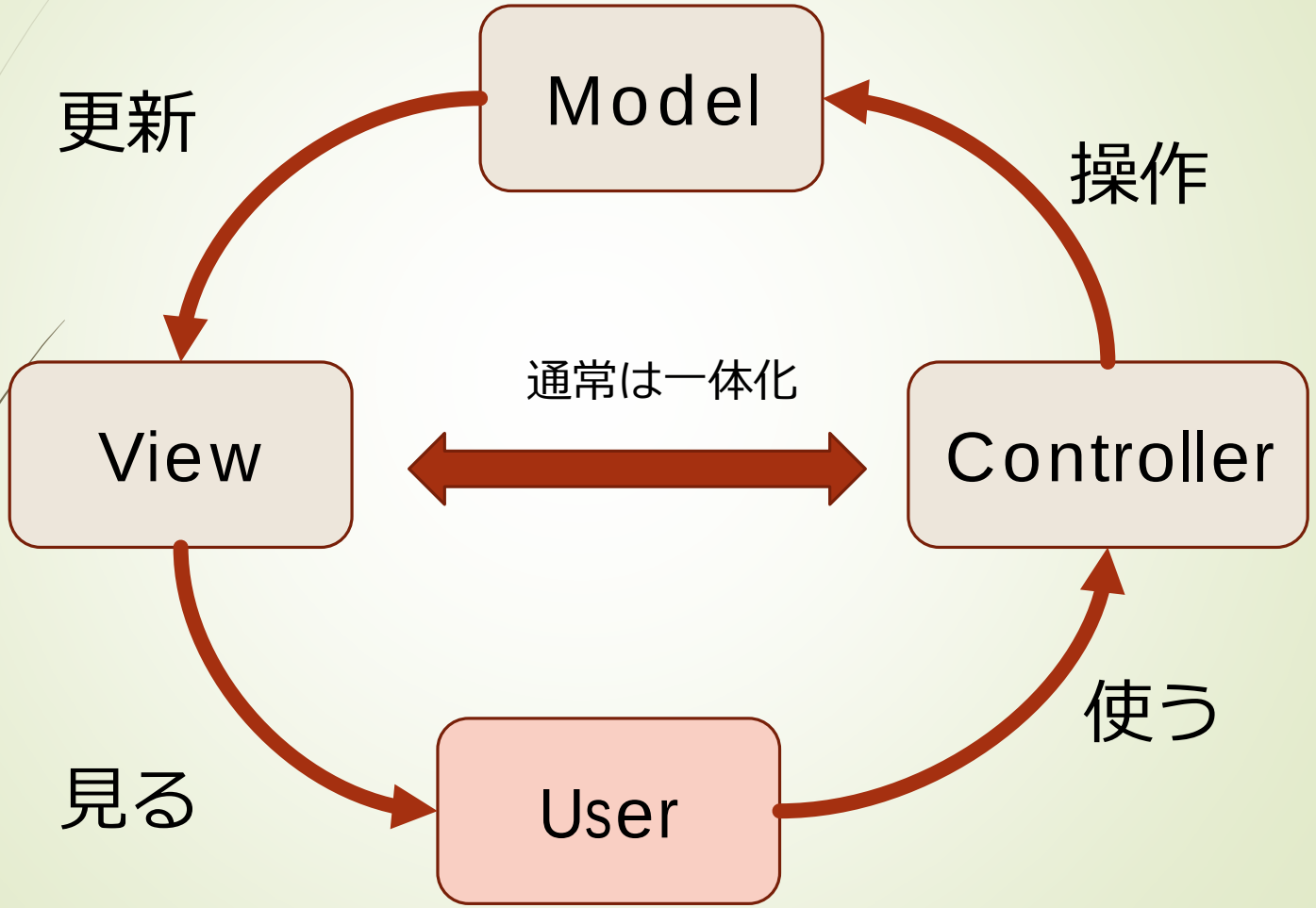
- java.util.EventListenerの派生インターフェイス
- イベント発生時に応答する
- イベントに対応したメソッドを追加
- イベントの種類に応じて、異なる処理を用意しても良い

イベントを発生させる側

- イベントリスナーのリストを保持
- イベントリスナーの登録メソッド
  - 削除メソッドも
- イベントが発生時
  - 全てのリスナーにイベントを通知

MVCという考え方

- Model
  - アプリケーションが対象とする模型
- View
  - GUIなどの表現
- Controller
  - GUIからModelを制御
- 三つの要素をできるだけ分離


Model
更新
操作
通常は一体化
View
Controller
見る
使う
User

イベント利用の例

- シミュレーションの例
  - シミュレーションモデル(M)とGUI(VとC)を適切に分離
  - MとGUIの間をイベントで通信
- マウスイベントを活用した例
  - マウスで絵を描く

シミュレーションとそのGUIを例に

- シミュレーション
  - 数理モデルとして記述されている
  - もともとGUIとは無関係
- 元のモデルに手を加えずに、シミュレーション結果の表示のためにGUIを作るのがベスト
- シミュレーションをGUIが観測

イベントとリスナーの定義

- イベント
  - 初期化と状態更新を通知

```
public class SimulationEvent extends EventObject {
    //イベントの種類
    public static enum EventType {
        INITIALIZED, //初期化
        UPDATED;//更新
    }
    private final EventType eventType;
    public SimulationEvent(Object source,EventType eventType) {
        super(source);        this.eventType = eventType;
    }
    public EventType getEventType() {return eventType; }
}
```

イベントとリスナーの定義

- イベントリスナー
  - 状態変化したときの処理を定義

```
public interface SimulationEventListener extends EventListener {

    public void stateChanged(SimulationEvent event);
}
```

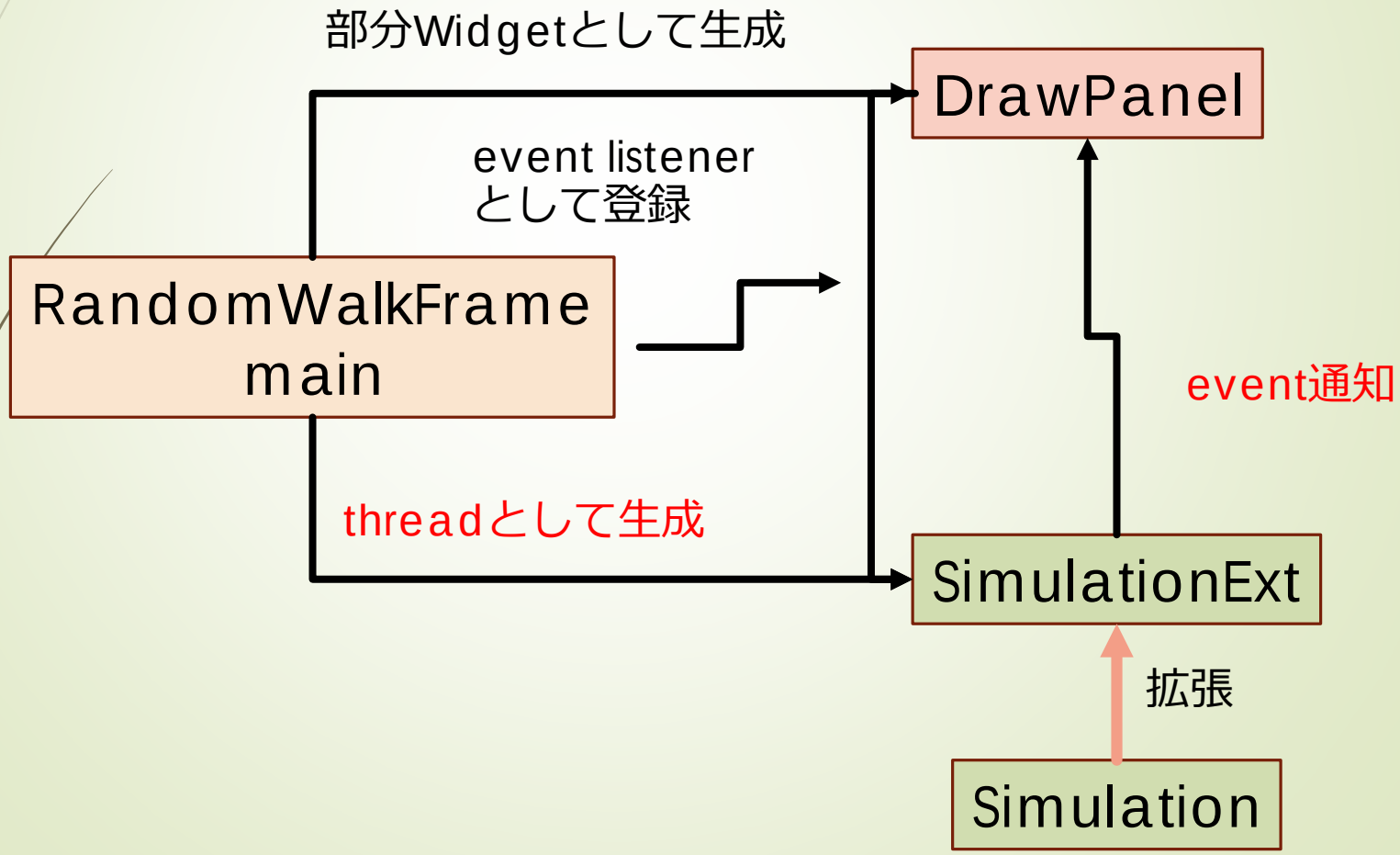
シミュレーションクラスの拡張 SimulationExt

- イベント発生
  - fireStateChangedメソッドによりイベント発生
- イベントリスナー登録
  - addSimulationEventListenerメソッド

GUI側

- DrawPanelクラス
  - イベントリスナーを実装
  - イベントが発生すると描画
- RandomWalkFrameクラス
  - イベントリスナーであるJPanelを登録

構成イメージ


部分Widgetとして生成
DrawPanel
event listener
として登録
RandomWalkFrame
main
event通知
threadとして生成
SimulationExt
拡張
Simulation

マルチスレッド

- シミュレーションとGUIが独立して動く
- シミュレーションで状態変化があると、GUIに伝え、GUI側で図を描き直す

SimulationExtクラス

- Simulationクラスの拡張クラス
  - 初期化および更新時にeventをfire
  - eventのリスナを登録

```
public class SimulationExt extends Simulation implements Runnable
```

すべてのイベントリスナに状態変更を通知する

```
protected void fireStateChanged(EventType eventType) {
    SimulationEvent e = new SimulationEvent(this, eventType);
    listeners.stream().forEach(p -> p.stateChanged(e));
}

@Override
public List<Integer> oneStep() {
    List<Integer> list = super.oneStep();
    fireStateChanged(EventType.UPDATED);
    return list;
}

@Override
public void initialize() {
    super.initialize();
    fireStateChanged(EventType.INITIALIZED);
}
```

リスナ登録

```
public SimulationExt(int n) {
      super(n);
      listeners = Collections.synchronizedList(new ArrayList<>());
   }

   public void addSimulationEventListener(SimulationEventListener o) {
      listeners.add(o);
   }

   public void removeSimulationEventListener(SimulationEventListener o) {
      listeners.remove(o);
   }

   public void clearSimulationEventListener() {
      listeners.clear();
   }
```

シミュレーション部分をthreadとして走らせる

- シミュレーションを自律的に走らせる
- Runnable インターフェイス
  - 別スレッドで実行
  - スレッドを起動すると、runメソッドを起動

Runnableインターフェイスの実装

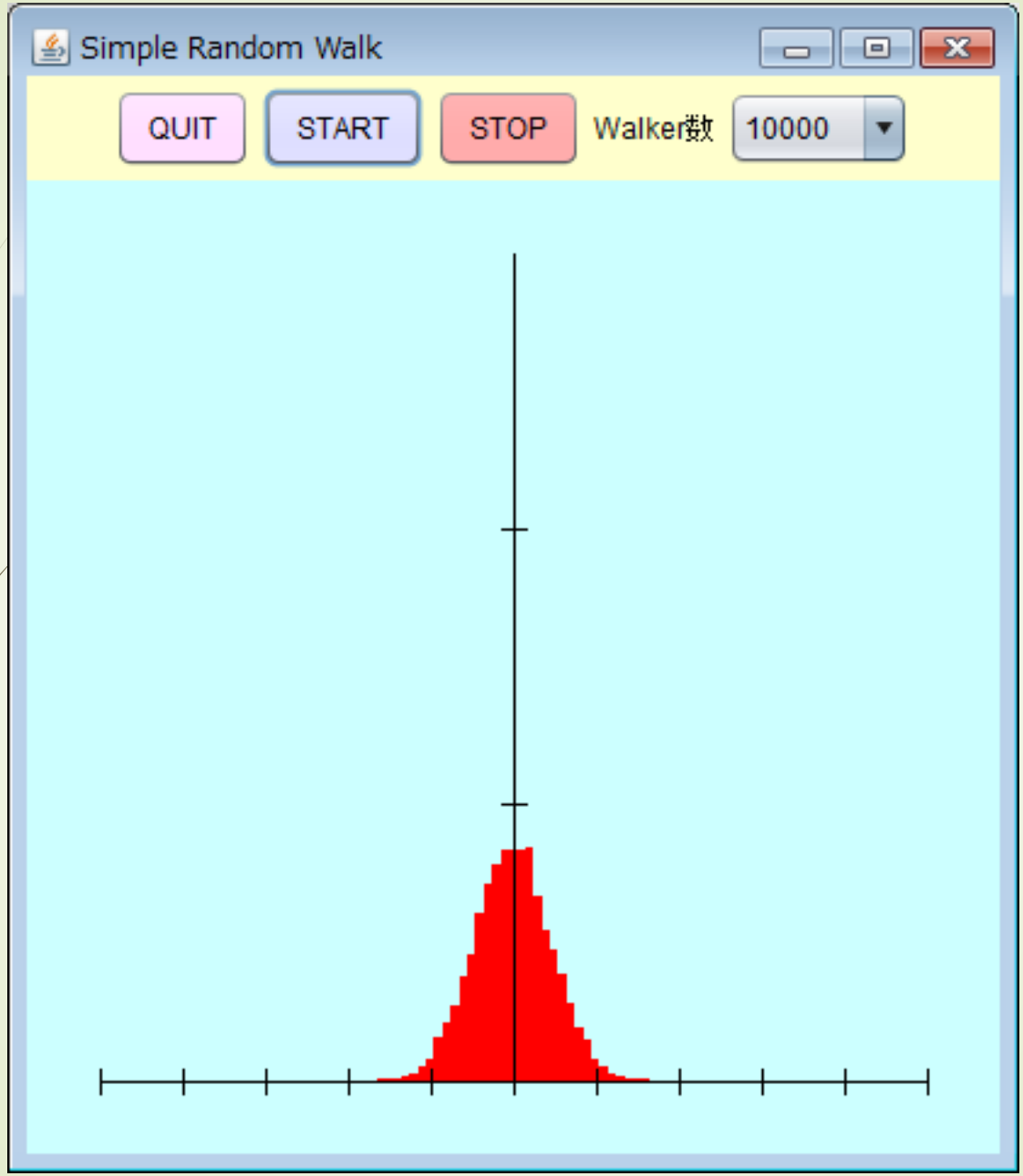
```
//無限ループを制御する変数の設定
public void start() { running = true;    t = 0;    }
public void stop() { running = false;    }

@Override
public void run() {
    while (running) {//無限ループ
        update(updateTiming);
        t++;
        if (t > tmax) {running = false;          }
        try {
            Thread.sleep(100);//100ms休む
        } catch (InterruptedException e) {
        }
    }
}
```

```
public class DrawPanel extends javax.swing.JPanel
    implements SimulationEventListener
```

```
@Override
  public void stateChanged(SimulationEvent event) {
    List<Point2D.Double> hist
        = PositionHistogram.getHist(simulation.getWalkers());
    initializeImage();
    createImage(hist);
    repaint();
  }
```

状態変化が起こったら、図を再構成して描画


Simple Random Walk
QUIT
START
STOP
Walker数
10000