独習Java第3版

14.1 代行イベントモデル
14.2 イベントクラス
14.3 イベントリスナ

14.1 代行イベントモデル(1/3)

- アプレットはGUIを提供する
- GUIベースのプログラムはイベントドリブンであり、コンソールアプリケーションはイベントドリブンでない

- イベントドリブンとは
  - ユーザや他のプログラムが実行した操作(イベント)に対応して処理を行なうプログラムの実行形式

代行イベントモデル(2/3)

- イベントドリブンインターフェイスを
  定義する方法は数種類あり、
  イベントモデルにより特定のメカニズムが決まる

- 代行イベントモデルとは
  - ユーザが操作を行った際に、ソースからその操作を示す
    オブジェクトのイベントを生成し、一連のリスナ(監視者)に
    送信する標準メカニズム

イベント

- イベントとは
  - ソースにおける状態変更を表すオブジェクト
  - ユーザーがGUI内の要素と対話（マウスのクリック等）を行った場合に生成される

ソース

- ソースとは
  - イベントを生成するもの
  - 以下の3つの責任を持つ
    - リスナが特定のタイプのイベントに関する通知を登録、解除できるようにするメソッドの提供
    - イベントの生成
    - 登録されている全てのリスナにイベントを送信することイベントは、単一のリスナに対するユニキャストでも、複数のリスナに対するマルチキャストでも構わない

イベントの登録、解除

- リスナがイベントを登録および解除できるようにするためソースが実装するメソッドは以下のとおり
  - addTypeListener()メソッド
    - public void addTypeListener(TypeListener *el*)
    - public void addTypeListener(TypeListener *el*) throws TooManyListenersException
  - removeTypeListener()メソッド
    - public void removeTypeListener(TypeListener *el*)

    - *el*はイベントリスナ

メソッドの説明

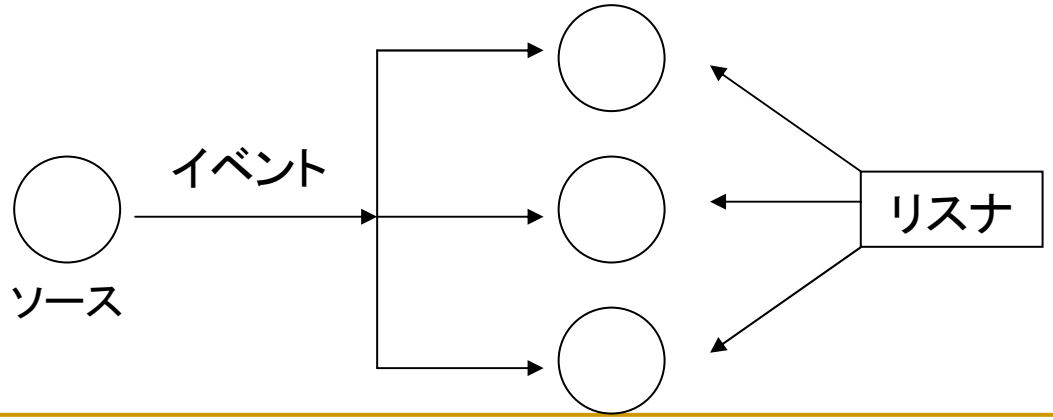
- 1つ目の構文は
  特定タイプのイベントリスナを複数登録
- 2つ目の構文は
  特定タイプのイベントリスナを一つ登録
- 3つ目の構文は
  特定タイプのイベント通知の登録を解除

リスナについて

- リスナには主に3つの責任がある
  - 特定のイベントに関する通知を受け取ることを登録すること
    （ソースの適切な登録メソッドを呼び出して行う）
  - そのタイプのイベントを受け取るインターフェイスを実装すること
  - その通知を受け取る必要がなくなった場合には登録を解除すること
    （登録解除はソースの適切な
    解除メソッドを呼び出して行う）

代行イベントモデル(3/3)

- 下の図は代行イベントモデルを示す
- ソースは一連のリスナにイベントを
  マルチキャストしている
- リスナはそのタイプのイベントに関する通知を受
  けるインターフェイスを実装する



14.2 イベントクラス

- 各種タイプのAWTイベントを表す一連のクラスが用意されている
- EventObjectコンストラクタ
  - EventObject(Object *src*)
    - *src*はイベントを生成するオブジェクト
    - このクラスには次の2つのメソッドがある
- getSource()メソッド、toString()メソッド
  - Object getSource()
  - String toString()

AWTEventクラス

- EventObjectを拡張したもので、java.awtパッケージに含まれている
- AWTイベントクラスは全てこのクラスのサブクラス
  - AWTEventコンストラクタ
    - AWTEvent(Object *source*, int *id*)
      - *source*：イベントを生成するオブジェクト
      - *id*：イベントのタイプ
  - getID()メソッド、toString()メソッド
    - int getID()
    - String toString()

java.awt.eventに含まれる主なイベントクラス

イベント	生成される条件
ActionEvent	ボタンを押したとき、リスト項目をダブルクリックしたとき、メニュー項目を選択したとき
AdjustmentEvent	スクロールバーを操作した時
ComponentEvent	コンポーネントが隠れた、移動した、サイズ変更された、表示可能になった時
ContainerEvent	コンポーネントがコンテナに追加した、またはコンテナから削除した時
FocusEvent	コンポーネントがキーボードフォーカスを取得した、または失った時
InputEvent	マウスイベントまたはキーイベントが発生したとき
ItemEvent	チェックボックスまたはリスト項目をクリックしたとき、押したとき、離したとき。 そのほか、マウスをコンポーネントに入れた、またはコンポーネントから出したとき
KeyEvent	キーボードから入力を受け取った時
MouseEvent	マウスをドラックまたは移動した時、クリックした時、押した時、話した時、その他、 マウスをコンポーネントに入れた、またはコンポーネントから出した時
TextEvent	テキストエリアまたはテキストフィールドの値を変更した時
WindowEvent	ウィンドウを活動化した時、閉じた時、非活動化した時、アイコン化解除した時、 開いた時、終了した時

ComponentEventクラス(1/2)

- AWTEventを拡張したクラス
- いくつかのタイプのコンポーネントイベントの識別に使用するint型定数が定義される

- ComponentEventコンストラクタ
  - Component Event(Component *src*, int *type*)
    - *src*：イベントを生成したオブジェクトの参照
    - *type*：イベントのタイプ

ComponentEventクラス(2/2)

- ComponentEventクラスの主な定数

定数	説明
COMPONENT_HIDDEN	コンポーネントが隠れた
COMPONENT_MOVED	コンポーネントが移動した
COMPONENT_RESIZED	コンポーネントのサイズが変更された
COMPONENT_SHOWN	コンポーネントが表示可能になった

- getComponent()メソッド
  - このイベントを生成したコンポーネントを返す
  - 構文
    Component getComponent()

InputEventクラス

- ComponentEventのサブクラスで、
  コンポーネント入力イベントのスーパークラス
- KeyEvent、MouseEventをサブクラスに持つ
- Inputに関する修飾子の情報を扱う場合に
  使用できるint型整数が定義されている

- getModifiers()メソッドはイベントの
  修飾子フラグを全て含むint型の値を返す
  - int getModifiers()

MouseEventクラス(1/3)

- InputEventのサブクラス
- マウスイベントのタイプ識別に使用できるint型定数が定義されている

定数	説明
MOUSE_CLICKED	マウスをクリックした
MOUSE_DRAGGED	マウスをドラッグした
MOUSE_ENTERED	マウスがコンポーネントに入った
MOUSE_EXITED	マウスがコンポーネントから出た
MOUSE_MOVED	マウスを移動した
MOUSE_PRESSED	マウスを押した
MOUSE_RELEASED	マウスを放した
MOUSE_WHEEL	マウスのホイールを回した

MouseEventクラス(2/3)

- 主なコンストラクタ
  - MouseEvent(Component *src*, int *type*, long *when*, int *modifiers*, int *x*, int *y*, int *clicks*, boolean *triggersPopup*)
    - *src*:生成したコンポーネントの参照
    - *type*:イベントのタイプ
    - *modifiers*:イベント発生時にどの修飾子が押されたか
    - *x,y*:マウス座標
    - *clicks*:クリック回数
    - *triggerPopup*:ポップアップメニュー表示の可否

MouseEventクラス(3/3)

- getX()メソッド、getY()メソッド
  - int getX()　　int getY()
    - イベント発生時のマウス座標を返す
- getPoint()メソッド
  - Point getPoint()
    - ソースコンポーネントを基準としてマウスイベントの場所を表すPointオブジェクトを返す

Pointクラス

- java.awtパッケージで定義されている
- publicなインスタンス変数のx,yがある
- translatePoint()メソッド
  - void translatePoint(int x,int y)
    - イベントの場所の変更
    - x,yはイベントの座標に追加される
- getClickCount()メソッド
  - int getClickCount()
    - イベント発生時のマウスクリック回数を返す

14.3 イベントリスナ

- java.util.EventListenerインターフェイスについて
  - イベントを処理するインターフェイスを指定するもので、定数やメソッドを定義するものではない
  - 全てのイベントリスナインターフェイスは このインターフェイスを拡張しなくてはいけない
  - インターフェイスは全てjava.awt.eventパッケージで宣言されている

AWTイベントクラスと リスナインターフェイス

イベントクラス	リスナインターフェイス
ActionEvent	ActionListener
AdjustmentEvent	AdjustmentListener
CompornentEvent	CompornentListener
ContainerEvent	ContainerListener
FocusEvent	FocusListener
ItemEvent	ItemListener
KeyEvent	KeyListener
MouseEvent	MouseListener,MouseMotionListener
MouseWheelEvent	MouseWheelListener
TextEvent	TextListener
WindowEvent	WindowListener

イベントの登録と解除

- マウスイベントの登録および解除するには Componentクラスのメソッドを用いる
  - それぞれのイベント通知を登録(通知可にする)
    - void addMouseListener(MouseListener *ml*)
    - void addMouseMotionListener(MouseMotionListener *mml*)
  - それぞれのイベント通知登録を解除
    - void removeMouseListener(MouseListener *ml*)
    - void removeMouseMotionListener(MouseMotionListener *mml*)
  - *ml*はマウスリスナ、*mml*はマウスモーションリスナ

MouseListenerインターフェイス

- マウスイベントを受けるメソッドを定義する
  - void mouseClicked(MouseEvent *me*)
  - void mouseEntered(MouseEvent *me*)
  - void mouseExited(MouseEvent *me*)
  - void mousePressed(MouseEvent *me*)
  - void mouseReleased(MouseEvent *me*)

MouseMotionListenerインターフェイス

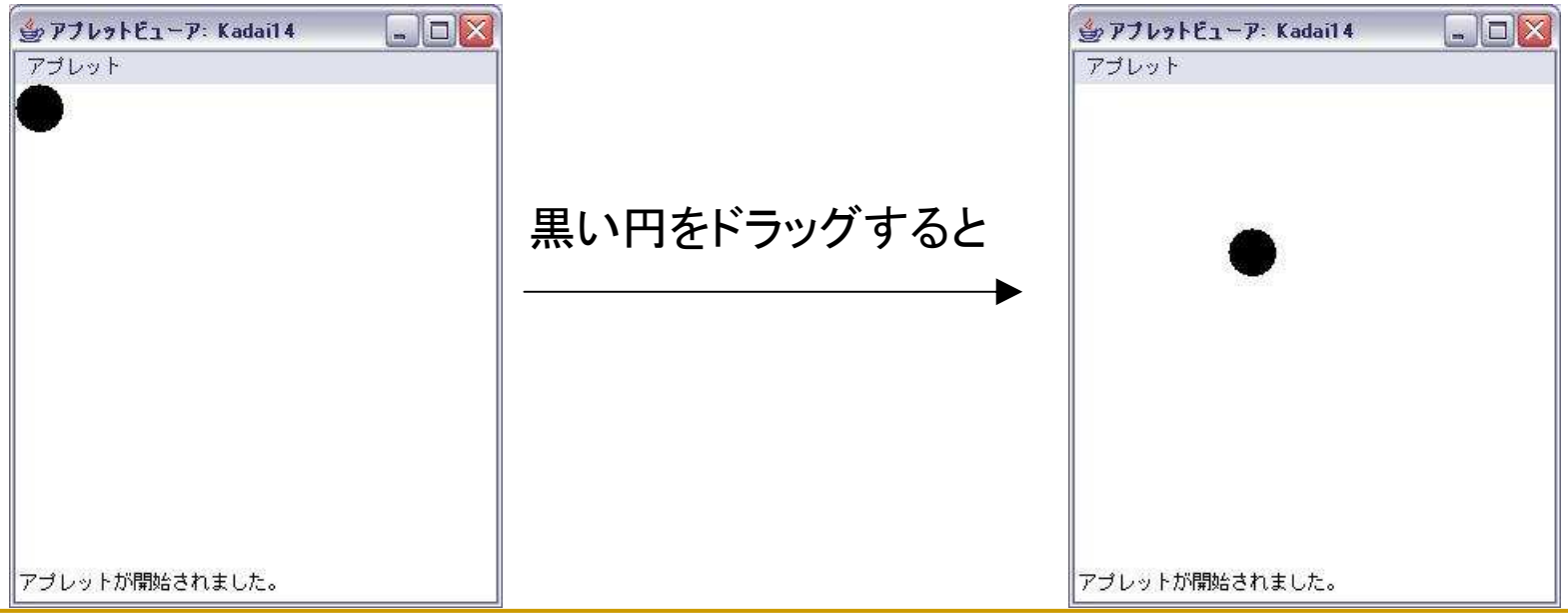
- 次のようなマウスイベントを受け取る
  2つのメソッドを定義している
  - mouseDragged()メソッド
    - void mouseDragged(MouseEvent *me*)
  - mouseMoved()メソッド
    - void mouseMoved(MouseEvent *me*)
  - *me*はソースで生成されたMouseEventオブジェクト
  - 登録されている全てのリスナの適切なメソッドが
    呼び出され、引数としてイベントが渡される

例

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
/*
  <applet code="MouseMotionEvents" width=300
      height=300>
  </applet>
*/
public class MouseMotionEvents extends Applet
implements MouseListener, MouseMotionListener
      {
  Point p;
  public void init() {
    addMouseListener(this);
    addMouseMotionListener(this);
  }
  public void mouseClicked(MouseEvent me) {
  }
public void mouseEntered(MouseEvent me) {
  }
public void mouseExited(MouseEvent me) {
  }
public void mousePressed(MouseEvent me) {
    p = me.getPoint();
    repaint();
  }
public void mouseReleased(MouseEvent me) {
    p = null;
    repaint();
  }
public void mouseDragged(MouseEvent me) {
    p = me.getPoint();
    repaint();
  }
public void mouseMoved(MouseEvent me) {
  }
public void paint(Graphics g) {
    if (p != null) {
      Dimension d = getSize();
      int xc = d.width / 2;
      int yc = d.height / 2;
      g.drawLine(xc, yc, p.x, p.y);
    }
  }
}
```

課題

- 画面上に小さい円を描き、それをドラッグすることで円を移動させられるプログラムを作りなさい
  - ただし、画面ちらつき防止のため
    ダブルバッファリングを用いなさい



アプレット実行直後

円を移動させた後