9 回目　メニューとポップアップメニュー

今日の講義で学ぶ内容

・メニューの利用
・メニューのカスタマイズ
・ポップアップメニュー

メニューの利用

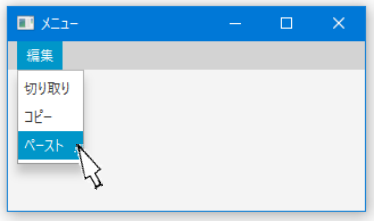
§1　メニューを配置してみましょう

メニューを用いることにより，欲しい機能をすばやく呼び出すことができます．

ソースファイル名：Sample9_1.java

```
// ※HP よりインポート文をここへ貼り付けてください

// メニューの配置
public class Sample9_1 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // メニューバーを生成します
      MenuBar mb = new MenuBar();
      mb.setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));

      // メニューを生成します
      Menu m = new Menu("編集");

      // メニュー項目を生成します
      MenuItem mi1 = new MenuItem("切り取り");
      MenuItem mi2 = new MenuItem("コピー");
      MenuItem mi3 = new MenuItem("ペースト");
      mi1.setId("cut");
      mi2.setId("copy");
      mi3.setId("paste");

      // メニューを組み立てます
      ObservableList<Menu> lstm = mb.getMenus();
      lstm.add(m);
      ObservableList<MenuItem> lstmi = m.getItems();
      lstmi.add(mi1);
      lstmi.add(mi2);
      lstmi.add(mi3);

      // イベントハンドラを設定します
```

```
        MyEventHandler actionhandler = new MyEventHandler();
        m.addEventHandler(ActionEvent.ANY, actionhandler);

        // レイアウト BorderPane を生成／設定します
        BorderPane bp = new BorderPane();
        bp.setTop(mb);

        // シーンを生成／設定します
        Scene scene = new Scene(bp);

        // ステージを設定します
        stage.setScene(scene);
        stage.setTitle("メニュー");

        // ステージを表示します
        stage.show();
    }

    // イベントハンドラ（イベント処理）クラスの宣言
    private class MyEventHandler implements EventHandler<ActionEvent>
    {
        public void handle(ActionEvent e)
        {
            MenuItem mi = (MenuItem)e.getTarget();
            System.out.println(mi.getId());
        }
    }

    public static void main(String[] args)
    {
        launch(args);
    }
}
```

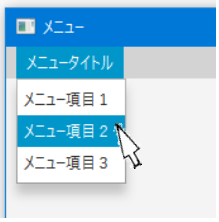


実行結果

```
copy      ← コピーを選択する
paste     ← ペーストを選択する
cut       ← 切り取りを選択する
  :
```

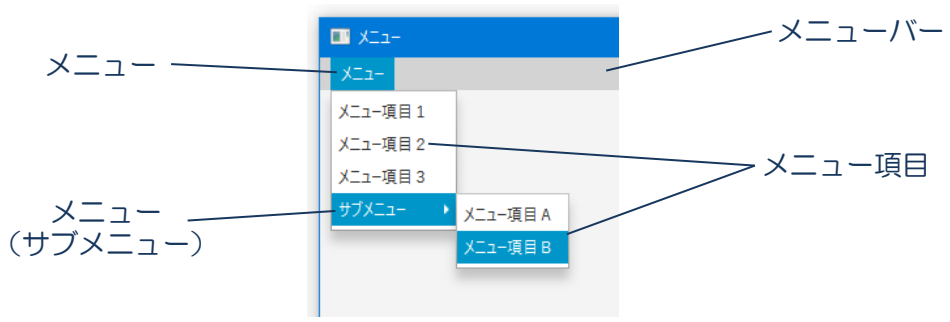
■メニューとは

クライアント領域の上部にある処理の選択を行う GUI 部品です．メニュータイトルをクリックすると，プルダウンメニューが表示され，この中から 1 つメニュー項目を選ぶことにより，処理を実行することができます．



■メニューの構成とクラス

メニューは，メニューバーとメニュー，メニュー項目から構成されます．メニューバーはいくつかのメニューを持ちます．メニューは，いくつかのメニュー項目またはメニュー（サブメニュー）をもちます．



それぞれ以下のクラスにより表現されます．

・メニューバー　　→ MenuBar クラス
・メニュー　　→ Menu クラス
・メニュー項目　　→ MenuItem クラス

■メニューバークラス MenuBar

メニューバーはクラス MenuBar により表現されます．各種設定を行うメソッドが準備されています．

・メニューバーの生成　　→ new MenuBar();
・背景色の指定　　→ setBackground(…(Color.LIGHTGRAY,…));

※ラベルの背景色の設定に用いたメソッドであり，Region クラスから継承したものと同じです．

背景が明るい灰色のメニューバーが生成されます．この他，メニューバーのサイズ指定や周囲の空白エリア指定ができます．

■メニュークラス Menu

メニューはクラス Menu により表現されます．各種設定を行うメソッドが準備されています．

・メニューの生成　　→ new Menu("編集");

タイトルが「編集」のメニューが生成されます．この他，アイコンの貼付けや無効化を行えます．

■メニュー項目クラス MenuItem

メニュー項目はクラス MenuItem により表現されます．各種設定を行うメソッドが準備されています．

・メニュー項目の生成　　→ new MenuItem("切り取り");
・識別子の設定（"cut"を識別子として）　→ setId("cut");
・識別子の取得（String 型）　　→ getId();

タイトルが「切り取り」のメニュー項目が生成され，識別子に"cut"が設定されます．この他，アイコン

の貼付けや無効化，ショートカットキーの割り当てなどができます．

■メニューの組み立て

まず，レイアウト BorderPane にメニューバーを配置します．

□レイアウト BorderPane にメニューバーを配置します

```
1. BorderPane bp = new BorderPane();
2. bp.setTop(mb);
```

※ボーダーペイン bp の上領域にメニューバーmb を配置します．

次に，レイアウトに GUI 部品を配置するのと同じように，メニューバーにメニューを，メニューにメニュー項目を配置していきます．

□メニューバーにメニューを配置します

```
1. ObservableList<Menu> lstm = mb.getMenus();
2. lstm.add(m);
```

※変数 mb は MenuBar クラスのオブジェクトとします．

※メニューバー mb からメニューリストを受け取ります．リストにメニュー m を追加します．

□メニューにメニュー項目を配置します

```
1. ObservableList<MenuItem> lstmi = m.getItems();
2. lstmi.add(mi);
```

※変数 m は Menu クラスのオブジェクト，変数 mi は MenuItem クラスのオブジェクトとします．

※メニュー m からメニュー項目リストを受け取り，リストにメニュー項目 mi を追加します．

■メニューとアクションイベント

メニュー項目をクリックするとアクションイベントが発生します．

■アクションイベントを処理するイベントハンドラインタフェース EventHandler<ActionEvent>

アクションイベントはイベントハンドラクラスで受け取り，対応する処理を行います．

1．EventHandler<ActionEvent>インタフェースを実装してイベントハンドラクラスを宣言

2．継承される void handle(ActionEvent e);メソッドをオーバーライドして処理を記述

※発生したイベントがメソッドの引数 e に渡されて呼び出されます

〔コード例〕

```
1. class MyEventHandler implements EventHandler<ActionEvent>{
2.   public void handle(ActionEvent e)
3.   {
```

```
4.      // ここにイベントに対応する処理を記述します
5.   }
6. }
```

■メニューへイベントハンドラを登録

GUI 部品やシーン，ステージは様々なイベントを発生します．メニュー項目から発生するイベントはメニューで受け取ることができます．メニューにイベントハンドラを登録します．

〔コード例〕

```
1. MyEventHandler eh = new MyEventHandler();
2. m.addEventHandler(ActionEvent.ANY, eh);
```

※オブジェクト eh をイベントハンドラとして Menu クラスのオブジェクト m に登録すると，この Menu クラスのオブジェクト m に関連づいている MenuItem クラスのオブジェクトで発生するイベントを受け取ることができます．

※イベントハンドラは MenuItem クラスのオブジェクトに個々に登録することもできます．

■メニュー項目と識別子

複数のメニュー項目を 1 つのイベントハンドラで処理するとき，イベントが発生したらどのメニュー項目から発生したのかを知る必要があります．メニュー項目には識別子を設定することができます．

MenuItem クラスに識別子の設定と取得を行うメソッドが準備されています．

- 識別子の設定（"cut"を識別子として）　→ `setId("cut");`
- 識別子の取得（String 型）　→ `getId();`

※ボタンなどの識別子の設定に用いたメソッドは Node クラスから継承されたものです．これらはクラス MenuItem で新たに宣言されたもので異なるものです．

■利用したクラスの一覧

MenuBar クラス←Control←Region←Parent←Node←Object	
`MenuBar(){…}`	メニューバーを生成します．
`void setBackground(Background v){…}`	背景の設定（色など）を v にします．
`ObservableList<Menu> getMenus(){…}`	メニューリストを取得します．
Menu クラス←MenuItem←Object	
`Menu(String t){…}`	タイトル t をもつメニューを生成します．
`ObservableList<MenuItem> getItems(){…}`	メニュー項目リストを取得します．
MenuItem クラス←Object	
`MenuItem(String t){…}`	タイトル t をもつメニュー項目を生成します．
`void setId(String s){…}`	文字列 s をメニュー項目の識別子に設定します．
`String getId(){…}`	メニュー項目の識別子を取得します．

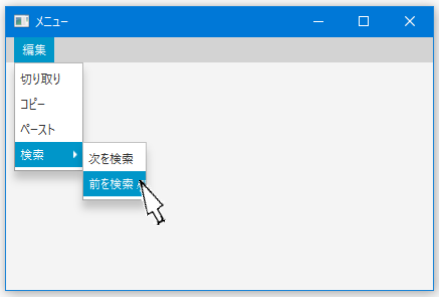
§2　サブメニューを配置してみましょう

メニュー項目を配置できる場所にさらにメニューを配置してサブメニューを構成できます.

ソースファイル名：Sample9_2.java

```
// ※HP よりインポート文をここへ貼り付けてください

// メニューの階層
public class Sample9_2 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // メニューバーを生成します
      MenuBar mb = new MenuBar();
      mb.setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));

      // メニューを生成します
      Menu m = new Menu("編集");
      Menu n = new Menu("検索");

      // メニュー項目を生成します
      MenuItem mi1 = new MenuItem("切り取り");
      MenuItem mi2 = new MenuItem("コピー");
      MenuItem mi3 = new MenuItem("ペースト");
      MenuItem mi4 = new MenuItem("次を検索");
      MenuItem mi5 = new MenuItem("前を検索");
      mi1.setId("cut");
      mi2.setId("copy");
      mi3.setId("paste");
      mi4.setId("next");
      mi5.setId("previous");

      // メニューを組み立てます
      ObservableList<Menu> lstm = mb.getMenus();
      lstm.add(m);
      ObservableList<MenuItem> lstmi = m.getItems();
      lstmi.add(mi1);
      lstmi.add(mi2);
      lstmi.add(mi3);
      lstmi.add(n);
      lstmi = n.getItems();
      lstmi.add(mi4);
      lstmi.add(mi5);

      // イベントハンドラを設定します
      MyEventHandler actionhandler = new MyEventHandler();
      m.addEventHandler(ActionEvent.ANY, actionhandler);
      n.addEventHandler(ActionEvent.ANY, actionhandler);

      // レイアウト BorderPane を生成／設定します
      BorderPane bp = new BorderPane();
```

```
        bp.setTop(mb);

        // シーンを生成／設定します
        Scene scene = new Scene(bp);

        // ステージを設定します
        stage.setScene(scene);
        stage.setTitle("メニュー");

        // ステージを表示します
        stage.show();
    }

    // イベントハンドラ（イベント処理）クラスの宣言
    private class MyEventHandler implements EventHandler<ActionEvent>
    {
        public void handle(ActionEvent e)
        {
            MenuItem mi = (MenuItem)e.getTarget();
            System.out.println(mi.getId());
        }
    }

    public static void main(String[] args)
    {
        launch(args);
    }
}
```



実行結果

```
next            ← 次を検索を選択する
previous        ← 前を検索を選択する
copy            ← コピーを選択する
  :
```

■サブメニューの構成

メニュー項目を配置する場所に，メニューを配置してサブメニューを構成することができます．このとき，サブメニューのメニュー項目で発生するイベントはサブメニューにイベントハンドラを登録して受け取ります．

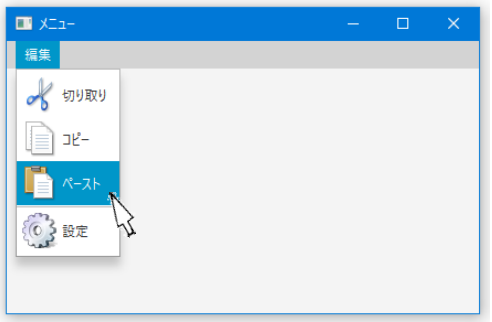
§3　メニュー項目にアイコンを配置してみましょう

ラベルやボタンと同じように，メニュー項目にアイコンを配置することができます．

ソースファイル名：Sample9_3.java

```
// ※HP よりインポート文をここへ貼り付けてください

// メニュー項目にアイコンを配置
public class Sample9_3 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // メニューバーを生成します
      MenuBar mb = new MenuBar();
      mb.setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));

      // メニューを生成します
      Menu m = new Menu("編集");

      // メニュー項目を生成します
      MenuItem mi1 = new MenuItem("切り取り");
      MenuItem mi2 = new MenuItem("コピー");
      MenuItem mi3 = new MenuItem("ペースト");
      MenuItem sp  = new SeparatorMenuItem();
      MenuItem mi4 = new MenuItem("設定");
      mi1.setId("cut");
      mi2.setId("copy");
      mi3.setId("paste");
      mi4.setId("setting");
      mi1.setGraphic(new ImageView("cut.png"));
      mi2.setGraphic(new ImageView("copy.png"));
      mi3.setGraphic(new ImageView("paste.png"));
      mi4.setGraphic(new ImageView("setting.png"));

      // メニューを組み立てます
      ObservableList<Menu> lstm = mb.getMenus();
      lstm.add(m);
      ObservableList<MenuItem> lstmi = m.getItems();
      lstmi.add(mi1);
      lstmi.add(mi2);
      lstmi.add(mi3);
      lstmi.add(sp);
      lstmi.add(mi4);

      // イベントハンドラを設定します
      MyEventHandler actionhandler = new MyEventHandler();
      m.addEventHandler(ActionEvent.ANY, actionhandler);
```

```
        // レイアウト BorderPane を生成／設定します
        BorderPane bp = new BorderPane();
        bp.setTop(mb);

        // シーンを生成／設定します
        Scene scene = new Scene(bp);

        // ステージを設定します
        stage.setScene(scene);
        stage.setTitle("メニュー");

        // ステージを表示します
        stage.show();
    }

    // イベントハンドラ（イベント処理）クラスの宣言
    private class MyEventHandler implements EventHandler<ActionEvent>
    {
        public void handle(ActionEvent e)
        {
            MenuItem mi = (MenuItem)e.getTarget();
            System.out.println(mi.getId());
        }
    }

    public static void main(String[] args)
    {
        launch(args);
    }
}
```



実行結果

```
copy          ← コピーを選択する
settings      ← 設定を選択する
paste         ← ペーストを選択する
  :
```

■メニュー項目にアイコンを配置するには

クラス MenuItem にアイコン（画像）を配置するメソッドが準備されています.

・アイコンの配置　　→ setGraphic(new ImageView("cut.png"));

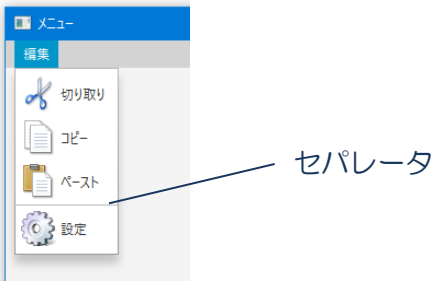
※ラベルのアイコンの設定に用いたメソッドは Labeled クラスから継承したものです. これらはクラス MenuItem で新たに宣言されたもので異なるものです.

※MenuItem クラスのサブクラスである Menu クラスでもアイコンを設定できます.

画像ファイル cut.png をメニュー項目にアイコンとして表示します.

■セパレータを配置するには

セパレータとは，メニュー項目をグループ化するための区切り線です．



セパレータは特別のメニュー項目であり，SeparatorMenuItem クラスで表現されます．通常のメニュー項目と同じようにメニュー項目リストに追加します．

・セパレータの生成　　→ new SeparatorMenuItem();

■利用したクラスの一覧

MenuItem クラス←Object

void setGraphic(Node n){…}	GUI 部品 n をメニュー項目に配置します． ※クラス Node はクラス ImageView のスーパークラスです．

SeparatorMenuItem クラス←CustomMenuItem←MenuItem←Object

SeparatorMenuItem(){…}	セパレータを生成します．

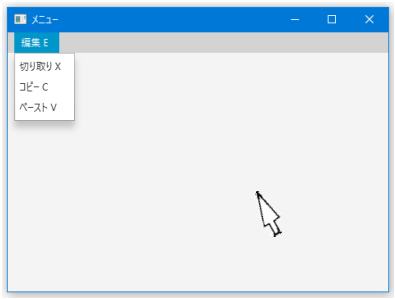
§4　メニュー項目にニーモニックを指定してみましょう

[Alt]+[A] や [Alt]+[X] でメニューやメニュー項目をクリックできるニーモニックを指定できます.

ソースファイル名：Sample9_4.java

```
// ※HP よりインポート文をここへ貼り付けてください

// メニュー項目にニーモニックを配置
public class Sample9_4 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // メニューバーを生成します
      MenuBar mb = new MenuBar();
      mb.setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));

      // メニューを生成します
      Menu m = new Menu("編集 _E");

      // メニュー項目を生成します
      MenuItem mi1 = new MenuItem("切り取り _X");
      MenuItem mi2 = new MenuItem("コピー _C");
      MenuItem mi3 = new MenuItem("ペースト _V");
      mi1.setId("cut");
      mi2.setId("copy");
      mi3.setId("paste");

      // メニューを組み立てます
      ObservableList<Menu> lstm = mb.getMenus();
      lstm.add(m);
      ObservableList<MenuItem> lstmi = m.getItems();
      lstmi.add(mi1);
      lstmi.add(mi2);
      lstmi.add(mi3);

      // イベントハンドラを設定します
      MyEventHandler actionhandler = new MyEventHandler();
      m.addEventHandler(ActionEvent.ANY, actionhandler);

      // レイアウト BorderPane を生成／設定します
      BorderPane bp = new BorderPane();
      bp.setTop(mb);

      // シーンを生成／設定します
      Scene scene = new Scene(bp);

      // ステージを設定します
      stage.setScene(scene);
      stage.setTitle("メニュー");
```

```
        // ステージを表示します
        stage.show();
    }

    // イベントハンドラ（イベント処理）クラスの宣言
    private class MyEventHandler implements EventHandler<ActionEvent>
    {
        public void handle(ActionEvent e)
        {
            MenuItem mi = (MenuItem)e.getTarget();
            System.out.println(mi.getId());
        }
    }

    public static void main(String[] args)
    {
        launch(args);
    }
}
```



実行結果

```
cut             ← [Alt]+[E] を押した後に [Alt]+[X]を押す
copy            ← [Alt]+[E] を押した後に [Alt]+[C]を押す
paste           ← [Alt]+[E] を押した後に [Alt]+[V]を押す
  :
```

■ニーモニックの有効化／無効化

ニーモニックは MenuItem クラスで指定できます．また，そのサブクラス（Menu クラスなど）でも指定できます．

クラス MenuItem にニーモニックを有効／無効化するメソッドが準備されています．ニーモニックはデフォルトで有効の状態です．

・ニーモニックの有効/無効の設定　　→ setMnemonicParsing(boolean value);
・ニーモニックの有効/無効の取得（boolean 型）→ isMnemonicParsing();

※ラベルのニーモニックの有効／無効化に用いたメソッドは Labeled クラスから継承したものです．これらはクラス MenuItem で新たに宣言されたもので異なるものです．

※MenuItem クラスのサブクラスである Menu クラスでもニーモニックの有効／無効化の設定ができます．

△現在，MenuBar 上に配置されるトップ Menu に対してニーモニックの有効／無効化の設定とその挙動が一致しない不具合が見られますが，今後改善されるでしょう．

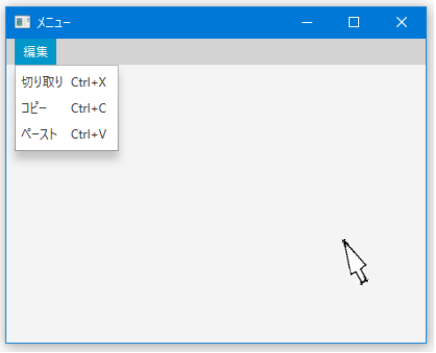
§5　メニュー項目にアクセラレータを指定してみましょう

[Ctrl]+[C] や [Ctrl]+[V] でメニュー項目をクリックできるアクセラレータを指定できます.

ソースファイル名：Sample9_5.java

```
// ※HP よりインポート文をここへ貼り付けてください

// メニュー項目にアクセラレータを配置
public class Sample9_5 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // メニューバーを生成します
      MenuBar mb = new MenuBar();
      mb.setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));

      // メニューを生成します
      Menu m = new Menu("編集");

      // メニュー項目を生成します
      MenuItem mi1 = new MenuItem("切り取り");
      MenuItem mi2 = new MenuItem("コピー");
      MenuItem mi3 = new MenuItem("ペースト");
      mi1.setId("cut");
      mi2.setId("copy");
      mi3.setId("paste");
      mi1.setAccelerator(KeyCombination.keyCombination("Ctrl+X"));
      mi2.setAccelerator(KeyCombination.keyCombination("Ctrl+C"));
      mi3.setAccelerator(KeyCombination.keyCombination("Ctrl+V"));

      // メニューを組み立てます
      ObservableList<Menu> lstm = mb.getMenus();
      lstm.add(m);
      ObservableList<MenuItem> lstmi = m.getItems();
      lstmi.add(mi1);
      lstmi.add(mi2);
      lstmi.add(mi3);

      // イベントハンドラを設定します
      MyEventHandler actionhandler = new MyEventHandler();
      m.addEventHandler(ActionEvent.ANY, actionhandler);

      // レイアウト BorderPane を生成／設定します
      BorderPane bp = new BorderPane();
      bp.setTop(mb);

      // シーンを生成／設定します
      Scene scene = new Scene(bp);

      // ステージを設定します
      stage.setScene(scene);
```

```
        stage.setTitle("メニュー");

        // ステージを表示します
        stage.show();
    }

    // イベントハンドラ（イベント処理）クラスの宣言
    private class MyEventHandler implements EventHandler<ActionEvent>
    {
        public void handle(ActionEvent e)
        {
            MenuItem mi = (MenuItem)e.getTarget();
            System.out.println(mi.getId());
        }
    }

    public static void main(String[] args)
    {
        launch(args);
    }
}
```



実行結果

```
cut             ← [Ctrl]+[X]を押す
copy            ← [Ctrl]+[C]を押す
paste           ← [Ctrl]+[V]を押す
  :
```

■アクセラレータとは

[Ctrl]+[C] や [Ctrl]+[V] でメニュー項目を直接クリックできます．これをアクセラレータといいます．

アクセラレータをメニュー項目に設定すると，その右側にキーコンビネーションが表示されます．

■アクセラレータを表現するクラス KeyCombination

アクセラレータのキーコンビネーションは KeyCombination クラスにより表現されます．様々なキーコンビネーションを生成するメソッドが準備されています．

・キーコンビネーションの生成　　→ keyCombination("Ctrl+X");　〔クラスメソッド〕

[Ctrl]+[X]のキーコンビネーションを生成します．また，引数には次のような文字列を指定できます．

・"Shift+"	Shift キーを押しながら…	・"Delete"	Delete キーを押す
・"Ctrl+"	Ctrl キーを押しながら…	・"Home"	Home キーを押す
・"Alt+"	Alt キーを押しながら…	・"End"	End キーを押す
・"F1",…	ファンクションキーF1 を押す	・"Page Up"	ページアップキーを押す
・"Esc"	エスケープキーを押す	・"Page Down"	ページダウンキーを押す
・"Tab"	タブキーを押す	・"Up"	矢印↑キーを押す
・"Space"	スペースキーを押す	・"Down"	矢印↓キーを押す
・"BackSpace"	BS キーを押す	・"Left"	矢印←キーを押す
・"Enter"	リターンキーを押す	・"Right"	矢印→キーを押す

■アクセラレータをメニュー項目に設定するには

クラス MenuItem にアクセラレータを設定するメソッドが準備されています．

・アクセラレータの設定　→ setAccelerator(KeyCombination.keyCombination("Ctrl+X"));

△クラス MenuItem のサブクラスである Menu クラスでもアクセラレータの設定がソフト的に可能ですが，表示されず動作もしません．この設定と挙動の不一致は今後解消されるでしょう．

[Ctrl]+[X]でこのメニュー項目がクリックされるように指定します．

■ニーモニックとアクセラレータの違いは?

・ニーモニックはそのメニュー項目が表示されたときに有効になります．

・アクセラレータはそのメニュー項目が表示されていないときでも有効です．

■利用したクラスの一覧

MenuItem クラス←Object

void setAccelerator(KeyCombination k){…}

キーコンビネーション k をアクセラレータに設定します．

KeyCombination クラス←Object

KeyCombination keyCombination(String s){…}　〔クラスメソッド〕

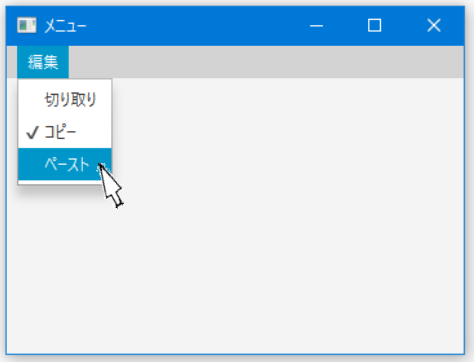
文字列 s が表すキーコンビネーションを生成します．

§6　チェックマークがつくメニュー項目にしてみましょう

チェックマークがつくメニュー項目を用いると，項目の ON／OFF を区別することができます．

ソースファイル名：Sample9_6.java

```
// ※HP よりインポート文をここへ貼り付けてください

// チェックマークがつくメニュー項目
public class Sample9_6 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // メニューバーを生成します
      MenuBar mb = new MenuBar();
      mb.setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));

      // メニューを生成します
      Menu m = new Menu("編集");

      // メニュー項目を生成します
      CheckMenuItem mi1 = new CheckMenuItem("切り取り");
      CheckMenuItem mi2 = new CheckMenuItem("コピー");
      CheckMenuItem mi3 = new CheckMenuItem("ペースト");
      mi1.setId("cut");
      mi2.setId("copy");
      mi3.setId("paste");

      // メニューを組み立てます
      ObservableList<Menu> lstm = mb.getMenus();
      lstm.add(m);
      ObservableList<MenuItem> lstmi = m.getItems();
      lstmi.add(mi1);
      lstmi.add(mi2);
      lstmi.add(mi3);

      // イベントハンドラを設定します
      MyEventHandler actionhandler = new MyEventHandler();
      m.addEventHandler(ActionEvent.ANY, actionhandler);

      // レイアウト BorderPane を生成／設定します
      BorderPane bp = new BorderPane();
      bp.setTop(mb);

      // シーンを生成／設定します
      Scene scene = new Scene(bp);

      // ステージを設定します
      stage.setScene(scene);
      stage.setTitle("メニュー");

      // ステージを表示します
```



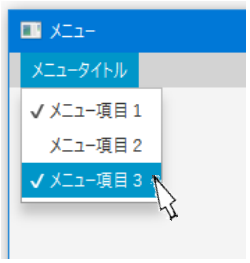
```
      stage.show();
   }

   // イベントハンドラ（イベント処理）クラスの宣言
   private class MyEventHandler implements EventHandler<ActionEvent>
   {
      public void handle(ActionEvent e)
      {
         CheckMenuItem mi = (CheckMenuItem)e.getTarget();
         System.out.println(mi.getId()+"/"+mi.isSelected());
      }
   }

   public static void main(String[] args)
   {
      launch(args);
   }
}
```

実行結果

```
copy/true      ← コピーをクリックし，チェックを入れる
paste/true     ← ペーストをクリックし，チェックを入れる
paste/false    ← ペーストをクリックし，チェックを外す
   :
```

■チェックマーク付きメニュー項目とは

メニュー項目にチェックマークをつけることができます．メニュー項目を選ぶたびに，チェックが入ったり外れたりします．チェックが入ったり外れたりするときに，アクションイベントが発生します．



■チェックマーク付きメニュー項目を管理するクラス CheckMenuItem

クラス CheckMenuItem はクラス MenuItem のサブクラスです．前で紹介した機能のほかに，チェック状態を設定したりその状態を取得したりするメソッドが準備されています．

・チェックマーク付きメニュー項目の生成　→ new CheckMenuItem("切り取り");
・チェック状態を設定（チェックを入れる）→ setSelected(true);
・チェック状態の取得（boolean 型）　→ isSelected();

CheckMenuItem クラス←MenuItem←Object

CheckMenuItem(String t){…}	タイトル t をもつチェックつきメニュー項目を生成します．
void setSelected(boolean b){…}	メニュー項目のチェック状態を設定します．
boolean isSelected(){…}	メニュー項目のチェック状態を取得します．

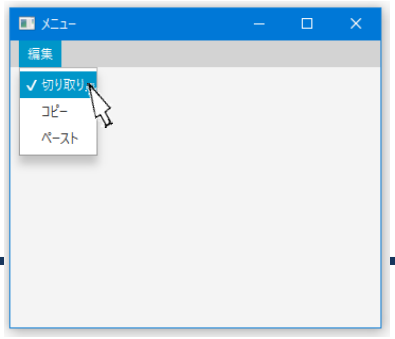
§7　排他的なチェックマークがつくメニュー項目にしてみましょう

チェックマークがつくメニュー項目を排他的に動作するようにできます.

ソースファイル名：Sample9_7.java

```
// ※HP よりインポート文をここへ貼り付けてください

// 排他的にチェックマークがつくメニュー項目
public class Sample9_7 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // メニューバーを生成します
      MenuBar mb = new MenuBar();
      mb.setBackground(new Background(new BackgroundFill(Color.LIGHTGRAY,null,null)));

      // メニューを生成します
      Menu m = new Menu("編集");

      // メニュー項目を生成します
      RadioMenuItem mi1 = new RadioMenuItem("切り取り");
      RadioMenuItem mi2 = new RadioMenuItem("コピー");
      RadioMenuItem mi3 = new RadioMenuItem("ペースト");
      mi1.setId("cut");
      mi2.setId("copy");
      mi3.setId("paste");

      // メニュー項目のグループを生成／設定します
      ToggleGroup tg = new ToggleGroup();
      mi1.setToggleGroup(tg);
      mi2.setToggleGroup(tg);
      mi3.setToggleGroup(tg);

      // メニューを組み立てます
      ObservableList<Menu> lstm = mb.getMenus();
      lstm.add(m);
      ObservableList<MenuItem> lstmi = m.getItems();
      lstmi.add(mi1);
      lstmi.add(mi2);
      lstmi.add(mi3);

      // イベントハンドラを設定します
      MyEventHandler actionhandler = new MyEventHandler();
      m.addEventHandler(ActionEvent.ANY, actionhandler);

      // レイアウト BorderPane を生成／設定します
      BorderPane bp = new BorderPane();
      bp.setTop(mb);

      // シーンを生成／設定します
      Scene scene = new Scene(bp);
```

```
        // ステージを設定します
        stage.setScene(scene);
        stage.setTitle("メニュー");

        // ステージを表示します
        stage.show();
    }

    // イベントハンドラ（イベント処理）クラスの宣言
    private class MyEventHandler implements EventHandler<ActionEvent>
    {
        public void handle(ActionEvent e)
        {
            RadioMenuItem mi = (RadioMenuItem)e.getTarget();
            System.out.println(mi.getId()+"/"+mi.isSelected());
        }
    }

    public static void main(String[] args)
    {
        launch(args);
    }
}
```



実行結果

```
copy/true       ← コピーをクリックし，チェックを入れる（その他のチェックは外れる）
cut/true        ← 切り取りをクリックし，チェックを入れる（その他のチェックは外れる）
paste/true      ← ペーストをクリックし，チェックを入れる（その他のチェックは外れる）
  :
```

■排他的にチェックマークがつくメニュー項目を作成するには

クラス RadioMenuItem と排他的に動作するメニュー項目の範囲を指定するクラス ToggleGroup を使用します．範囲の指定の方法はラジオボタンと同じです．

■排他的にチェックマークつくメニュー項目を管理するクラス RadioMenuItem

クラス RadioMenuItem はクラス MenuItem のサブクラスです．前で紹介した機能のほかに，
チェック状態を設定したりその状態を取得したりするメソッドが準備されています．

・チェックマーク付きメニュー項目の生成　→ new RadioMenuItem("切り取り");
・チェック状態を設定（チェックを入れる）→ setSelected(true);
・チェック状態の取得（boolean 型）　→ isSelected();
・トグルグループの設定　→ setToggleGroup(ToggleGroup v);

※ラジオボタンのトグルグループの設定に用いたメソッドは ToggleButton クラスから継承したものです．これらはクラス RadioMenuItem で新たに宣言されたもので異なるものです．

RadioMenuItem クラス←MenuItem←Object

`RadioMenuItem(String t){…}`	タイトル t をもつチェックつきメニュー項目を生成します.
`void setSelected(boolean b){…}`	メニュー項目のチェック状態を設定します.
`boolean isSelected(){…}`	メニュー項目のチェック状態を取得します.
`void setToggleGroup(ToggleGroup v) {…}`	クラス ToggleGroup の同じオブジェクトを持つもの同士でトグルグループを設定します.

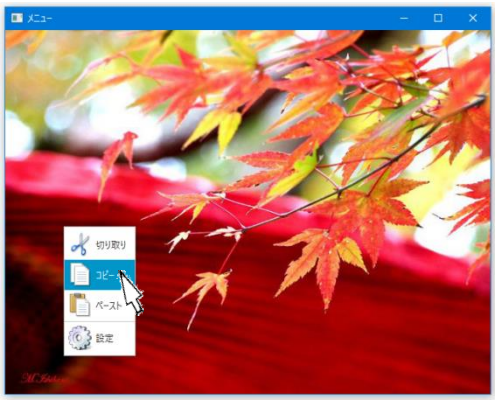
§8　ポップアップメニューを作成してみましょう

GUI 部品を右クリックすると表示されるポップアップメニューを作成できます.

ソースファイル名：Sample9_8.java

```
// ※HP よりインポート文をここへ貼り付けてください

// ポップアップメニューの作成
public class Sample9_8 extends Application
{
   public void start(Stage stage) throws Exception
   {
      // ポップアップメニューを生成します
      ContextMenu m = new ContextMenu();

      // メニュー項目を生成します
      MenuItem mi1 = new MenuItem("切り取り");
      MenuItem mi2 = new MenuItem("コピー");
      MenuItem mi3 = new MenuItem("ペースト");
      MenuItem sp  = new SeparatorMenuItem();
      MenuItem mi4 = new MenuItem("設定");
      mi1.setId("cut");
      mi2.setId("copy");
      mi3.setId("paste");
      mi4.setId("setting");
      mi1.setGraphic(new ImageView("cut.png"));
      mi2.setGraphic(new ImageView("copy.png"));
      mi3.setGraphic(new ImageView("paste.png"));
      mi4.setGraphic(new ImageView("setting.png"));

      // ポップアップメニューを組み立てます
      ObservableList<MenuItem> lstmi = m.getItems();
      lstmi.add(mi1);
      lstmi.add(mi2);
      lstmi.add(mi3);
      lstmi.add(sp);
      lstmi.add(mi4);

      // イベントハンドラを設定します
      MyEventHandler actionhandler = new MyEventHandler();
      m.addEventHandler(ActionEvent.ANY, actionhandler);

      // ラベルの生成とポップアップメニューの登録
      Label lb=new Label();
      lb.setGraphic(new ImageView("colorleaves.jpg"));
      lb.setContextMenu(m);

      // レイアウト VBox を生成／設定します
      VBox vb = new VBox();
```

```
        ObservableList<Node> lst = vb.getChildren();
        lst.add(lb);

        // シーンを生成／設定します
        Scene scene = new Scene(vb);

        // ステージを設定します
        stage.setScene(scene);
        stage.setTitle("メニュー");

        // ステージを表示します
        stage.show();
    }

    // イベントハンドラ（イベント処理）クラスの宣言
    private class MyEventHandler implements EventHandler<ActionEvent>
    {
        public void handle(ActionEvent e)
        {
            MenuItem mi = (MenuItem)e.getTarget();
            System.out.println(mi.getId());
        }
    }

    public static void main(String[] args)
    {
        launch(args);
    }
}
```



実行結果

```
copy          ← 画像を右クリック後，コピーをクリックする
paste         ← 画像を右クリック後，ペーストをクリックする
paste         ← 画像を右クリック後，ペーストをクリックする
 :
```

■ポップアップメニューとは

GUI 部品に関連付けることができるメニューです．GUI 部品の上でマウスを右クリックするとその位置にメニューが表示されます．メニュー項目をクリックすると，アクションイベントが発生します．

■ポップアップメニューを管理するクラス ContextMenu

クラス Menu の代わりにクラス ContextMenu を使用するとポップアップメニューを構成できます．メニュー項目はこれまで通りにクラス MenuItem を用いて生成・設定します．

・ポップアップメニューの生成　　→ new ContextMenu();

・メニュー項目リストの取得　　→ getItems();

※クラス ContextMenu は Window クラスのサブクラスに位置し，ウィンドウの一種です．また，ラベルで学習したツールチップで用いたクラス Tooltip と兄弟クラスです．

※クラス ContextMenu は，メニューを構成するためのメソッド getItems()を新たに宣言しています．

■ポップアップメニューの組み立て

メニューにメニュー項目を配置するのと同じ手順で行います．

```
1. ObservableList<MenuItem> lstmi = m.getItems();
2. lstmi.add(mi);
```

※変数 m は ContextMenu クラスのオブジェクト，変数 mi は MenuItem クラスのオブジェクトとします．

※ポップアップメニュー m からメニュー項目リストを受け取り，メニュー項目 mi を追加します．

■GUI 部品にポップアップメニューを登録するには?

クラス Control を拡張したクラス（Button，CheckBox，RadioButton，Label,…）にはポップアップメニューを登録するメソッドがあります．

・ポップアップメニューの登録　　→ setContextMenu(m);

クラス ContextMenu のオブジェクト m をポップアップメニューとして登録します．

■利用したクラスの一覧

ContextMenu クラス←PopupControl←PopupWindow←Window←Object

ContextMenu(){…}　　ポップアップメニューを生成します．

Label クラス←Labeled←Control←Region←Parent←Node←Object

void setContextMenu(ContextMenu c){…}　ポップアップメニューc を登録ます．