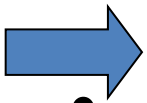
プログラミング入門2

第12回　インタフェースとクラス階層

講義資料について

- 新しい言語の機能(オブジェクト指向の機構)については、随時参考書などを参照するのがよい。
- 過去の資料も参考になる。
  - http://java2005.cis.k.hosei.ac.jp/
  - インタフェース　19回
  - 例外 21, 22回

テーマ：インタフェースとクラス階層



- インタフェース（概要）
- クラス階層とクラスObject
- 例外(概要)
- 演習

インタフェース

抽象クラス（インスタンスを生成しないクラス）をより抽象化したものに、**インタフェース**がある。

```
public interface インタフェース名 {
   型名　定数名=定数値;
   …
   返り値の型　メソッド名（引数）;
   …
}
```

インタフェースは次のような役割と特徴を持つ。
- 機能の表面的な名前だけをひとまとめに定義しておく場所
- 機能の内容（プログラム）は持てない。メソッドは、全て抽象メソッド。
  すなわち、インスタンスは作成できない。

インタフェースの実例

```
public interface Fillable {
   public void fill() ;
}
```

インタフェースFillableは、fillメソッドを持つ。これは中身を持たない抽象メソッドである。（abstractと宣言しなくて良い）

インタフェースの実装

インタフェースで定義されたメソッドの実体は、インタフェースを組み込と宣言したクラスの中に定義される。これを「インタフェースを実装する」という。

インタフェースを組み込むクラスは「implements インタフェース名」と宣言する。

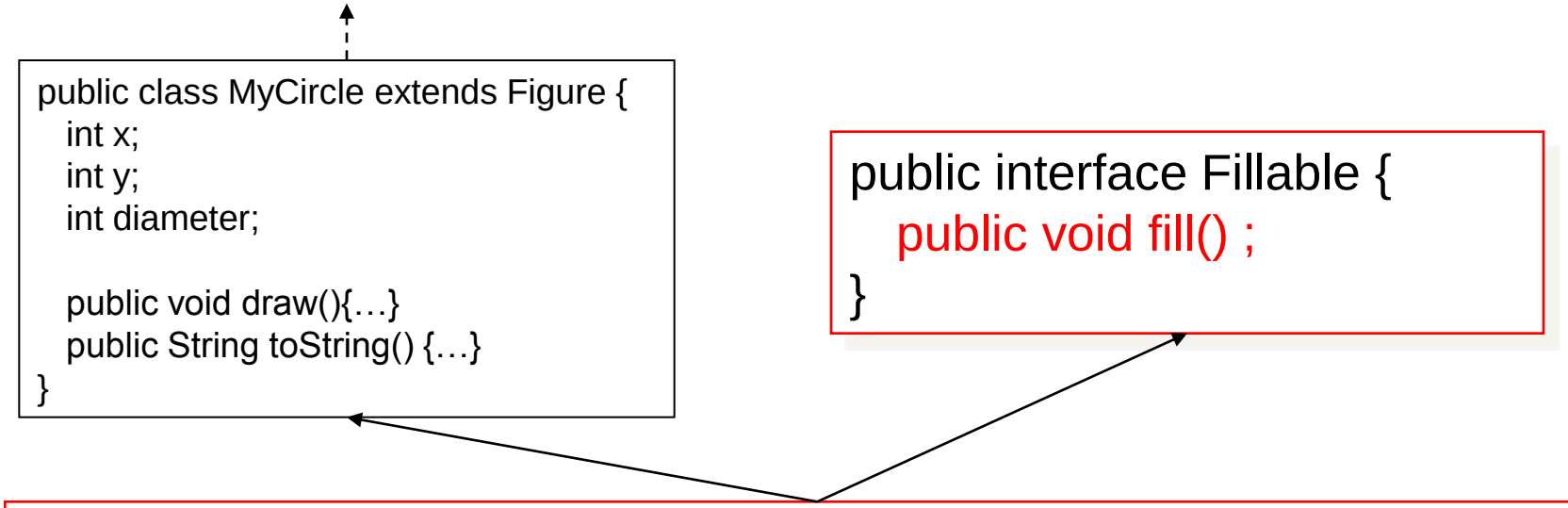
```
public class クラス名 implements インタフェース1, インタフェース2, ...
{
    ….
}
```

インタフェースとともに、スーパークラスを宣言する場合は下記の形で宣言する。

```
public class クラス名 extends 親クラス implements インタフェース1, インタフェース2, ...
{
    ……
}
```

インタフェース（例）

Fillableを組み込んだColoredCircleクラスを定義する。fill()メソッドを必ず実装しなければならない。インタフェースはそれを実装するクラスが特定のメソッドを持つことを保証する。



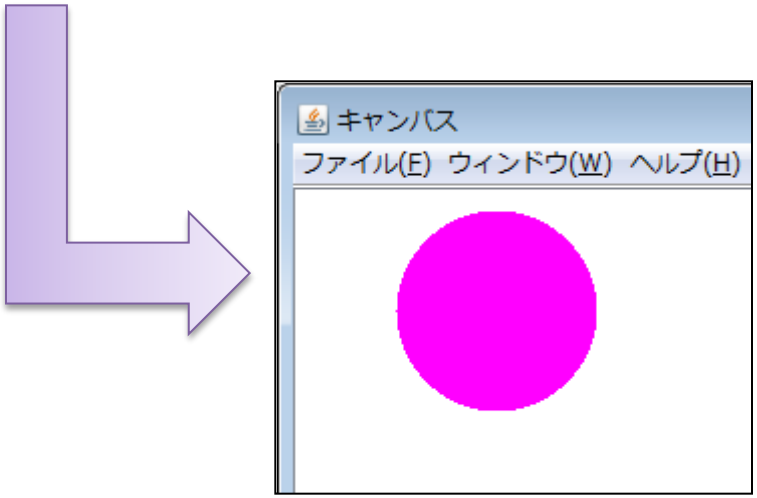

```
public class ColoredCircle extends MyCircle implements Fillable {
   int red; int green, int blue;
    public void draw() {  … }
   public void fill() {
        Canvas.setColor(red, green, blue);
        super.draw(); }
}
```

例題11より (Ex11FillTest)

インタフェースFillableを定義すると、Fillable型も同時に定義される。

```
1  package j2.lesson12;
2
3  import gpjava.Canvas;
4
5  public class Ex11FillTest {
6
7      public static void main(String[] args) {
8          Fillable f1 = new ColoredCircle(50, 10, 100, 255, 0, 255);
9          Canvas.show();
10         f1.fill();
11     }
12 }
```

ColoredCircleクラスは、Fillable型も持つ。Fillable型を保持するインスタンスでは、fillメソッドが必ず実行可能になっていることがポイント。



この例ではインタフェースを利用する利点はあまりないが、以降で述べるように複数のクラスが同じインタフェースを実装する場合に、効果を発揮する。

インタフェース（例）（共通のインタフェース）

インタフェースFillableを実装する、
クラスColoredRectangleを定義。

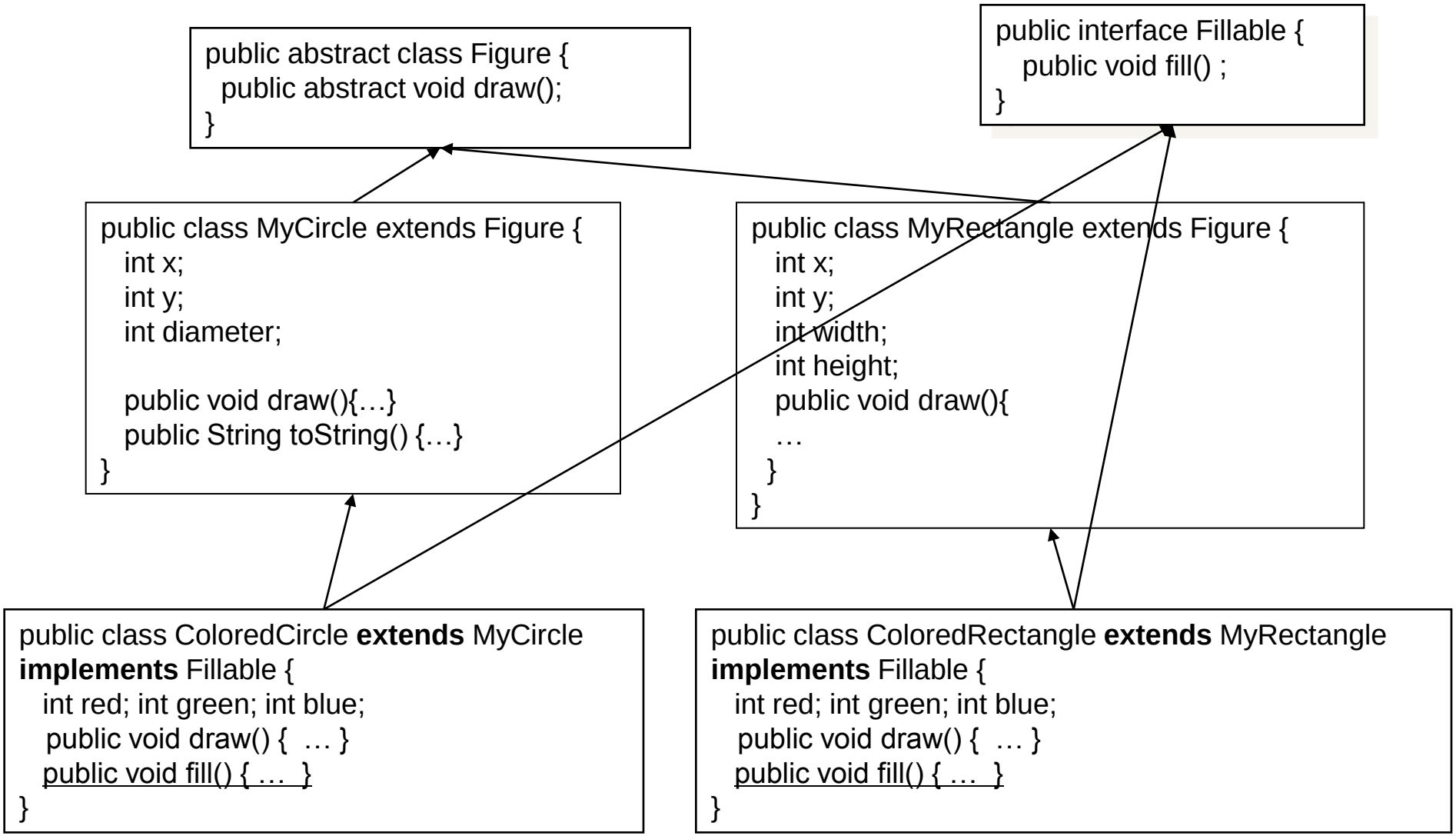
```
public abstract class Figure {
  public abstract void draw();
}
```

```
public class MyRectangle extends Figure {
   int x;
   int y;
   int width;
   int height;

   public void draw(){…}
}
```

```
public interface Fillable {
   public void fill() ;
}
```

```
public class ColoredRectangle extends MyRectangle implements Fillable
{
   int red; int green; int blue;
   public void draw() {  … }
   public void fill() {
      Canvas.setColor(red, green, blue);
      super.draw();
  }
}
```

共通のインタフェース



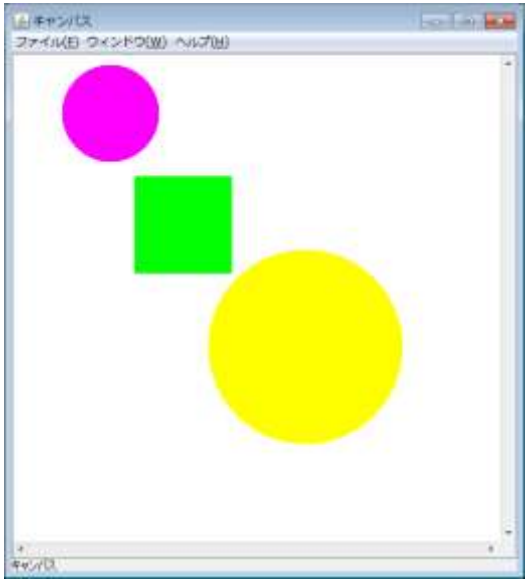
クラスColoredCircleおよびColoredRectangleは共通のインタフェースFillableをもとに実装される。両者ともfillメソッドを実装している。

例題13 (Ex13FillTest)

```
1  package j2.lesson12;
2
3  import gpjava.Canvas;
4
5  public class Ex13FillTest {
6
7      public static void main(String[] args) {
8          Fillable[] fs = {
9                  new ColoredCircle(50, 10, 100, 255, 0, 255),
10                 new ColoredRectangle(125, 125, 100, 100, 0, 255, 0),
11                 new ColoredCircle(200, 200, 200, 255, 255, 0)
12         };
13         Canvas.show();
14         for(int i = 0; i < fs.length; i++) {
15             Fillable fig = fs[i];
16             fig.fill();
17         }
18     }
19 }
```

Figure型の配列では対処できない。クラスFigureにはfillメソッドが定義されていないため。

インタフェースFillableを実装するクラスのインスタンスでは、fillメソッドが必ず実行可能になっていることがポイント。



テーマ：インタフェースとクラス階層

- インタフェース（概要）
- クラス階層とクラスObject
- 例外(概要)
- 演習

クラス階層

クラス間の継承関係は、図のような階層構造になる

```
public abstract class Figure extends Object {
  public abstract void draw();
}
```

（インタフェースは省略）

```
public class MyCircle extends Figure {
   int x;
   int y;
   int diameter;

   public void draw(){…}
   public String toString() {…}
}
```

```
public class MyRectangle extends Figure {
   int x;
   int y;
   int width;
   int height;
   public void draw(){
   …
  }
}
```

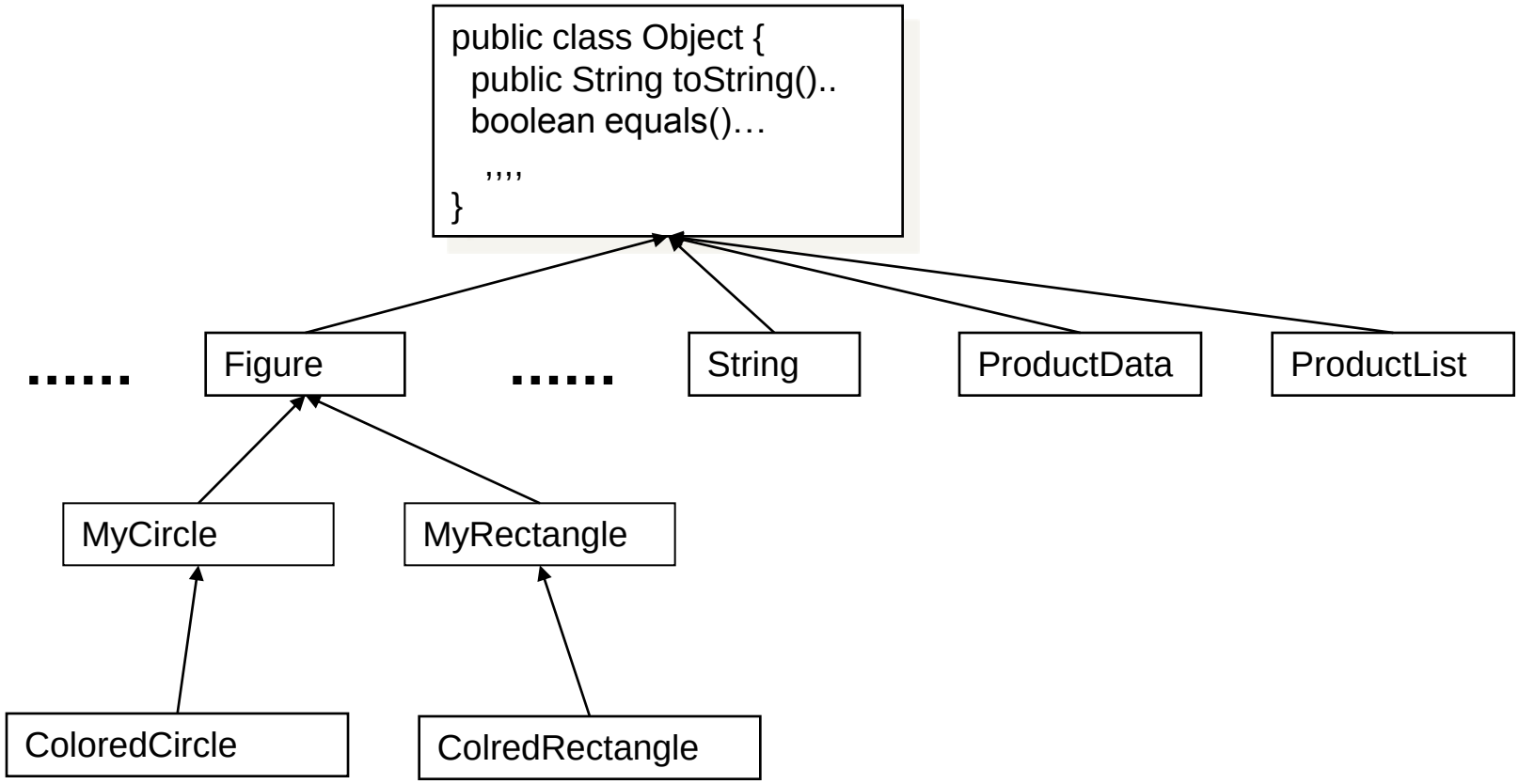
```
public class ColoredCircle extends MyCircle
implements Fillable {
   int red; int green; int blue;
   public void draw() {  … }
   public void fill() { …  }
}
```

```
public class ColoredRectangle extends MyRectangle
implements Fillable {
   int red; int green; int blue;
   public void draw() {  … }
   public void fill() { …  }
}
```

クラスObject(1)

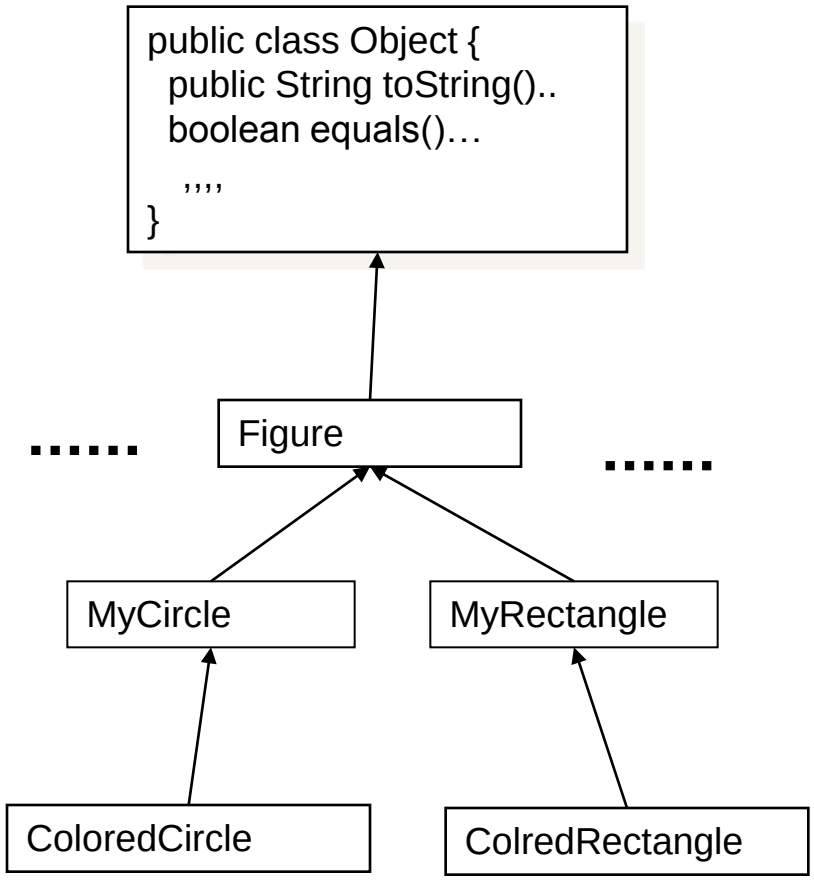
Java言語では、階層構造のルート（根）に、クラスObjectがある。

スーパークラスを指定しないクラスも、暗黙にクラスObjectを継承する。



クラスObject(2)

クラスObjectは、全てのインスタンスの最も基本的な構造を定義している。
よって、どのクラスのインスタンスもObjectクラスで定義された性質をもつ。



Objectには、次のメソッドが、すでに定義されている。（他にもいくつかある。）

```
equals()
toString()
```

あらゆるインスタンスは、このメソッドをあらかじめ備えることになる。

クラスObjectとオブジェクト指向

- インスタンスのことをオブジェクトと呼ぶことがある。
- Java言語は、インスタンス（オブジェクト）を中心にしてプログラミングを行う仕組みを備えており、このような仕組みを備えた言語は「**オブジェクト指向言語**」と呼ばれる。

Javaは、インスタンス（オブジェクト）に関して下記の重要な機構（メカニズム）を有する。

1. インスタンスを生成するための機構
   ＝「クラスからインスタンスが生成される」
2. インスタンスが主体となって、計算を進める機構。（メッセージパッシングによって計算が進む。）
   ＝「インスタンスがメソッドを持つ」

3. 多様なインスタンスを効率的に生成したり、整理するための機構
   ＝「継承によって種類の異なるインスタンスを効率的に作成する。」
   ＝「継承によってクラス階層構造が構築される」

クラスObjectとオブジェクト指向

前スライドの1,2,3と合わせて、 Java言語は、複数の人々が大規模なプログラムを効率よく開発するための機構を備える。

- インスタンスの型、抽象クラス、インタフェース
- パッケージ
- アクセスコントロール

学習の順番としては、以下が良い。
a． 前スライドの1,2（インスタンスと、インスタンスメソッド）の概念をよく知る。
b． 前スライドの3（クラスと継承）をよく知る。

その後に、
c． 型、抽象クラス、インタフェース等の知識を習得していく

テーマ:インタフェースとクラス階層

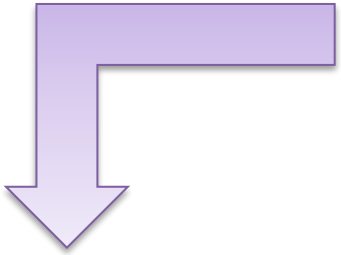
- インタフェース（概要）
- クラス階層とクラスObject
- 例外(概要)
- 演習

例外（概要）

「例外（Exception）」とは、プログラムの実行している途中に、
何らかの原因で想定していた処理が続けられない場合などに
発生するシグナル（警告メッセージ）のようなものである。

例題31より

0で割る割り算を行ったときどのような例外が発生するか確認せよ。



```
1  package j2.lesson12;
2
3  public class Ex31ExceptionTest {
4      public static void main(String[] args) {
5          System.out.println(2 / 0);
6      }
7  }
```

```
<終了> Ex31ExceptionTest [Java アプリケーション] C:¥usr¥Java¥jre6¥bin¥javaw.exe (2011/04/01 19:35:38)
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at j2.lesson12.Ex31ExceptionTest.main(Ex31ExceptionTest.java:5)
```

例外（概要） 例外を捕らえる(catch)

「発生」した例外は、プログラム中で「捕らえる」ことができる。

- シグナルに応じて処理を部分的に取りやめる、やり直させるなどの緊急回避的な処理が可能となり、プログラムの実行を続けることが可能となる。

```
1  package j2.lesson12;
2
3  public class Ex31ExceptionTest_1 {
4      public static void main(String[] args) {
5          try {
6              System.out.println(2 / 0);
7          } catch (ArithmeticException ex) {
8              System.out.println("0で割りました");
9          }
10     }
11 }
```

```
<終了> Ex31ExceptionTest_1 [Java アプリケーション] C:¥usr¥Java¥jre6¥bin¥javaw.ex
0で割りました
```

例外（概要） プログラムの中断

「発生」した例外をプログラム内では「捕らえない」場合は、
- プログラムのそのものの実行を中止する

ことになる。

右の例では、
プログラムが
5行目で中断している。

6行目以降は実行されていない。

```
1  package j2.lesson13.sample;
2
3  public class ExceptionTest {
4      public static void main(String[] args) {
5          int x = 2 / 0;
6          int y = x * 10;
7          System.out.println(y);
8      }
9
10 }
```

```
<終了> ExceptionTest (1) [Java アプリケーション] C:¥Program Files¥Java¥jdk1.6.0_12¥bin¥javaw.exe (2009.
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at j2.lesson13.sample.ExceptionTest.main(ExceptionTest.java:5)
```

例外とデバッグ

```
1  package j2.lesson13.sample;
2
3  public class ExceptionTest2 {
4      public static void main(String[] args) {
5          int a = 5 - 5;
6          int x = 2 / a;
7          int y = x * 10;
8          System.out.println(y);
9      }
10
11 }
```

```
<終了> ExceptionTest2 (1) [Java アプリケーション] C:¥Program Files¥Java¥jdk1.6.0_12¥bin¥javaw.exe (20
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at j2.lesson13.sample.ExceptionTest2.main(ExceptionTest2.java:6)
```

コンソールのメッセージを理解することは重要。どこで例外が発生したかが分かる。
普段プログラムをするときにちゃんと読むことを心がけるとよい。

上記では、ExceptionTest2クラスのmain関数の6行目で、何かおかしなことが起こってプログラムが途中で終了したことがわかる。
おかしなこととは、/ by zero （0による割り算）であることがわかる。

このメッセージを手がかりに、プログラムの6行目をまず探すと2 / aとある。
最終的に、5行目のaの値が0になることが原因であることが分かる。

例外(概要) 例外の種類

必ずしもプログラム中で捕らえなくて良い例外
＝ クラスRuntimeExceptionおよびそれを継承するクラスの例外

RuntimeException, NullPointerException
ArrayIndexOutOfBoundsException,
NumberFormatException,
ArithmeticException

必ずプログラム中で捕らえなくてはいけない例外
＝ クラスRuntimeExceptionを継承しないクラスの例外

IOException, InterruptedException

例外(概要) 例外の送出

```
1  package j2.lesson12;
2
3  import gpjava.Canvas;
4
5  public class MyCircle extends Figure {
6      int x;
7      int y;
8      int diameter;
9
10     public  MyCircle(int x, int y, int d) {
11         this.x = x;
12         this.y = y;
13         this.diameter = d;
14     }
15
16     public String toString() {
17         String result = "[";
18         result += "x:" + this.x;
19         result += " y:" + this.y;
20         result += " diameter:" + this.diameter;
21         result += "]";
22         return result;
23     }
24
25     public void draw() {
26         Canvas.drawOval(this.x, this.y, this.diameter, this.diameter);
27     }
28
29     //Ex32
30     public void testThrow() {
31         throw new RuntimeException("実験:実行時エラー");
32     }
33 }
```

例外の送出は
throw new 例外クラス(引数);
で行う。例外を「投げる」とも言う。

例題32(ExceptionTest)

```java
1  package j2.lesson12;
2
3  public class Ex32ExceptionTest {
4      public static void main(String[] args) {
5          MyCircle c = new MyCircle(50, 10, 50);
6          c.testThrow();
7      }
8  }
```

```java
29     //Ex32
30     public void testThrow() {
31         throw new RuntimeException("実験:実行時エラー");
32     }
```

```
<終了> Ex32ExceptionTest [Java アプリケーション] C:¥usr¥Java¥jre6¥bin¥javaw.exe (2011/04/01 20:10:58)
Exception in thread "main" java.lang.RuntimeException: 実験:実行時エラー
        at j2.lesson12.MyCircle.testThrow(MyCircle.java:31)
        at j2.lesson12.Ex32ExceptionTest.main(Ex32ExceptionTest.java:6)
```

RuntimeExceptionのようにプログラムで捕捉しなくて良い例外は、呼び出し元xで捕捉されなければ、さらにxの呼び出し元y、そこでもダメならさらにその元… というふうに順番に呼び出し元に例外が送られる。

テーマ：インタフェースとクラス階層

- インタフェース（概要）
- クラス階層とクラスObject
- 例外(概要)
- 演習

まとめ：インタフェースとクラス階層

- インタフェース（概要）
- クラス階層とクラスObject
- 例外(概要)
- 演習

本日の例題と問題

- Fillable インタフェースの演習
  - Ex11, Ex12, Ex13, Q11
- toString
  - Ex21
- 例外検出
  - Ex31, Ex32
- ItemList, ItemData
  - Q21, Q22, (Q23*), (Ex41*), (Q31*)

(Ex:例題, Q:問題, *は少し手間のかかる問題)

各自に適した順番で解けばよいが、上記の順番が自然な流れとなるよう構成されている。

例題集

パッケージ「j2.lesson12」を作成する。

パッケージやクラスの作成，実行の仕方の説明は省略する。
作り方を忘れた場合は過去のスライドや
http://java2010.cis.k.hosei.ac.jp/01/material-01/
を参考にせよ

例題のダウンロード

例題1では、あらかじめ次のプログラムを用意したので、ダウンロードして、利用して良い。(Figures.zipを解凍する)

Figure.java
MyCircle.java
MyRectangle.java

■例題11

問題:次の抽象メソッドfill()を保持するインタフェースFillableを作成せよ。さらに、
Fillableを実装するクラスとしてColoredCircleを作成しなおし、挙動を確かめよ。

Fillable抽象メソッド
public void fill();

ColoredCircleのスーパークラス: MyCircle インタフェース: Fillable

ColoredCircleインスタンス変数

変数の型と名前	初期値	説明
int red, green, blue	無し	色のRGB値(例: red=255,green=0 ,blue=255)

ColoredCircleインスタンスメソッド

返り値の型	メソッド名(引数)	機能
void	`draw()`	キャンバスにColoredCircleインスタンスを描画する
void	**fill()**	キャンバスにColoredCircleインスタンスを描画する(ただし、内部もcolorで塗りつぶされているものとする)

コンストラクタ
ColoredCircle(int x, int y, int d, int red, int green, int blue)

(クラス名: Fillable,ColoredCircle)
動作確認クラス: Ex11FillTest

例題11 MyCircle(ColoredCircleのスーパークラスとして必要)

```java
package j2.lesson12;

import gpjava.Canvas;

public class MyCircle extends Figure {
    int x;
    int y;
    int diameter;

    public  MyCircle(int x, int y, int d) {
        this.x = x;
        this.y = y;
        this.diameter = d;
    }

    public String toString() {
        String result = "[";
        result += "x:" + this.x;
        result += " y:" + this.y;
        result += " diameter:" + this.diameter;
        result += "]";
        return result;
    }

    public void draw() {
        Canvas.drawOval(this.x, this.y, this.diameter, this.diameter);
    }
}
```

Figure(MyCircleのスーパークラスとして必要)

```java
package j2.lesson12;

public abstract class Figure {
    public abstract void draw();

}
```

例題11(Fillable)

```
1  package j2.lesson12;
2
3  public interface Fillable {
4      public void fill();
5  }
```

例題11(ColoredCircle)

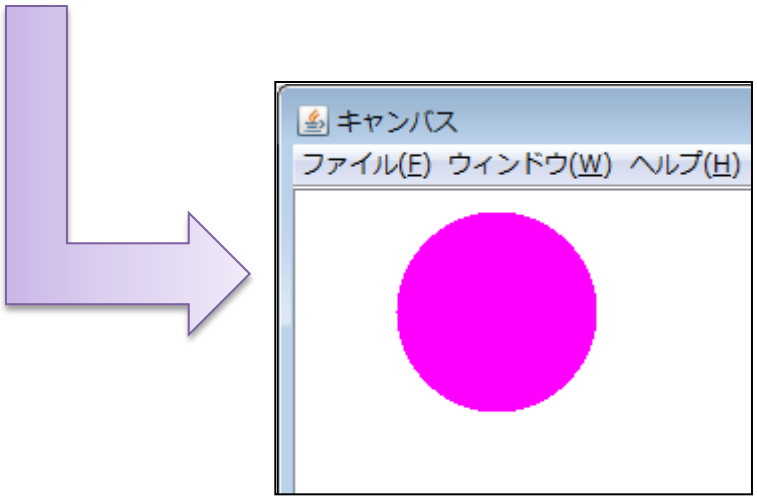
```
1  package j2.lesson12;
2
3  import gpjava.Canvas;
4
5  public class ColoredCircle extends MyCircle implements Fillable {
6      int red;
7      int green;
8      int blue;
9
10     public ColoredCircle(int x, int y, int d, int red, int green, int blue) {
11         super(x, y, d);
12         this.red = red;
13         this.green = green;
14         this.blue = blue;
15     }
16
17     public void fill() {
18         Canvas.setColor(red, green, blue);
19         Canvas.fillOval(x, y, diameter, diameter);
20     }
21
22     public void draw() {
23         Canvas.setColor(red, green, blue);
24         super.draw();
25     }
26 }
```

```
17     j2.lesson12.Fillable.fill を実装します
18         Canvas.setColor(red, green,
19         Canvas.fillOval(x, y, diam
20     }
```

ためしに下記をプログラムした後にfillメソッドを消し、
そのときに表示されるEclipseのメッセージを理解せよ。

例題11
(Ex11FillTest)

```java
package j2.lesson12;

import gpjava.Canvas;

public class Ex11FillTest {

    public static void main(String[] args) {
        Fillable f1 = new ColoredCircle(50, 10, 100, 255, 0, 255);
        Canvas.show();
        f1.fill();
    }
}
```



■例題12

問題：Fillableを実装するColoredRectangleクラスを作成し、挙動を確かめよ。

ColoredRectangleスーパークラス: MyRectangle　インタフェース：Fillable

ColoredRectangleインスタンス変数

変数の型と名前	初期値	説明
int red, green, blue	無し	色のRGB値(例: red=255, green=0, blue=255)

ColoredRectangleインスタンスメソッド

返り値の型	メソッド名(引数)	機能
void	`draw()`	キャンバスにColoredRectangleインスタンスを描画する
void	**fill()**	キャンバスにColoredRectangleインスタンスを描画する(ただし、内部もcolorで塗りつぶされているものとする)

コンストラクタ(ColoredCircle)
ColoredRectangle(int x, int y, int width, int height, int red, int green, int blue)

(クラス名： ColoredRectangle)
動作確認クラス： Ex12FillTest

例題12 MyRectangle
(ColoredRectangleのスーパークラスとして必要)

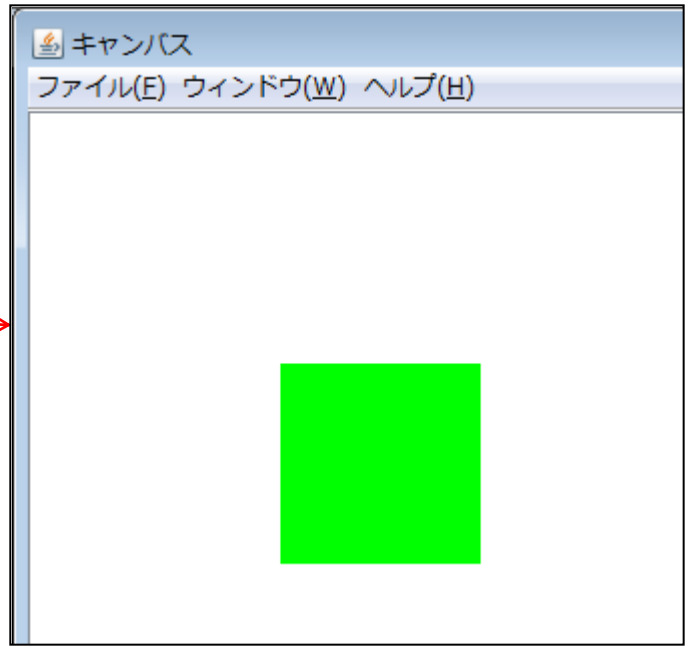
```java
package j2.lesson12;

import gpjava.Canvas;

public class MyRectangle extends Figure{
    int x;
    int y;
    int width;
    int height;

    public MyRectangle(int x, int y, int width, int height) {
        super();
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
    }

    public void draw() {
        Canvas.drawRect(x, y, width, height);
    }
}
```

例題12
(ColoredRectangle)

```java
package j2.lesson12;

import gpjava.Canvas;

public class ColoredRectangle extends MyRectangle implements Fillable {
    int red;
    int green;
    int blue;

    public ColoredRectangle(int x, int y, int width, int height, int red, int green, int blue) {
        super(x, y, width, height);
        this.red = red;
        this.green = green;
        this.blue = blue;
    }

    public void draw() {
        Canvas.setColor(red, green, blue);
        Canvas.drawRect(x, y, width, height);
    }

    public void fill() {
        Canvas.setColor(red, green, blue);
        Canvas.fillRect(x, y, width, height);
    }
}
```

例題12(Ex12FillTest)

```java
package j2.lesson12;

import gpjava.Canvas;

public class Ex12FillTest {

    public static void main(String[] args) {
        Fillable f2 = new ColoredRectangle(125, 125, 100, 100, 0, 255, 0);
        Canvas.show();
        f2.fill();
    }
}
```



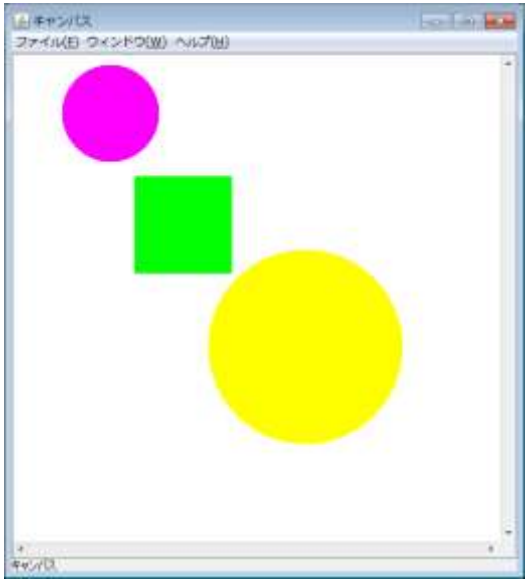
■例題13

問題：Fillable型の配列に、ColoredRectangle, ColoredCircleのインスタンスをいくつか追加し、それぞれ表示させよ。

Ex13FillTest

例題13
(Ex13FillTest)

```
1  package j2.lesson12;
2
3  import gpjava.Canvas;
4
5  public class Ex13FillTest {
6
7      public static void main(String[] args) {
8          Fillable[] fs = {
9                  new ColoredCircle(50, 10, 100, 255, 0, 255),
10                 new ColoredRectangle(125, 125, 100, 100, 0, 255, 0),
11                 new ColoredCircle(200, 200, 200, 255, 255, 0)
12         };
13         Canvas.show();
14         for(int i = 0; i < fs.length; i++) {
15             Fillable fig = fs[i];
16             fig.fill();
17         }
18     }
19 }
```



■例題21

問題:次のプログラムを実行し、コンソールの出力を確認せよ。

```
1  package j2.lesson12;
2
3  public class Ex21ToString {
4      public static void main(String[] args) {
5
6          Figure fig = new MyCircle(50, 10, 100);
7
8          String s = fig.toString();
9          System.out.println(s);
10
11         Figure fig2 = new ColoredCircle(150, 150, 100, 255, 0, 255);
12
13         String s2 = fig2.toString();
14         System.out.println(s2);
15
16         Object o = new Object();
17         String s3 = o.toString();
18         System.out.println(s3);
19     }
20 }
```

（クラス名： Ex21ToString）

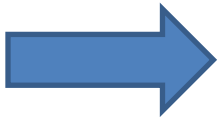
例題21(Ex21ToString)

```java
package j2.lesson12;

public class Ex21ToString {
    public static void main(String[] args) {
        Figure fig = new MyCircle(50, 10, 100);

        String s = fig.toString();
        System.out.println(s);

        Figure fig2 = new ColoredCircle(150, 150, 100, 255, 0, 255);

        String s2 = fig2.toString();
        System.out.println(s2);
    }
}
```

```
<終了> Ex21ToString [Java アプリケーション]
[x:50 y:10 diameter:100]
[x:150 y:150 diameter:100]
```

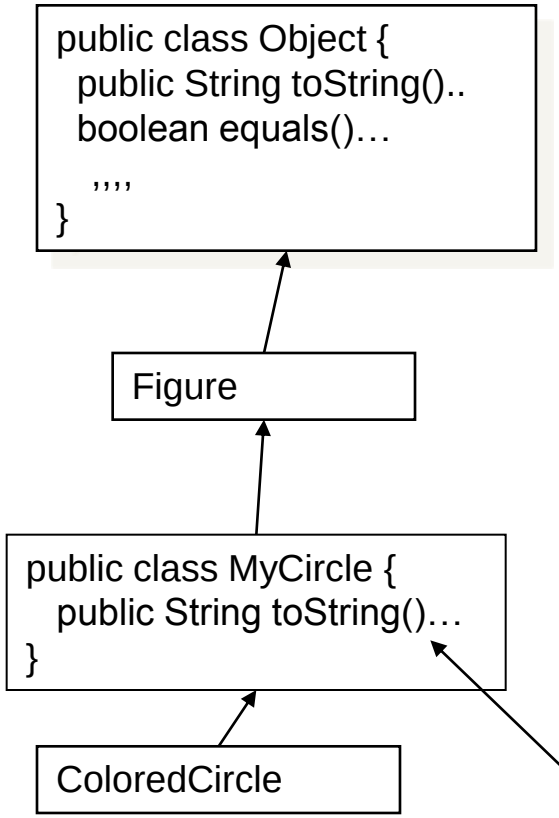
次にMyCircleクラスのtoString()をコメントアウトしてEx21ToStringを実行してみよ

```java
    /*
    public String toString() {
        String result = "[";
        result += "x:" + this.x;
        result += " y:" + this.y;
        result += " diameter:" + this.diameter;
        result += "]";
        return result;
    }
    */
```

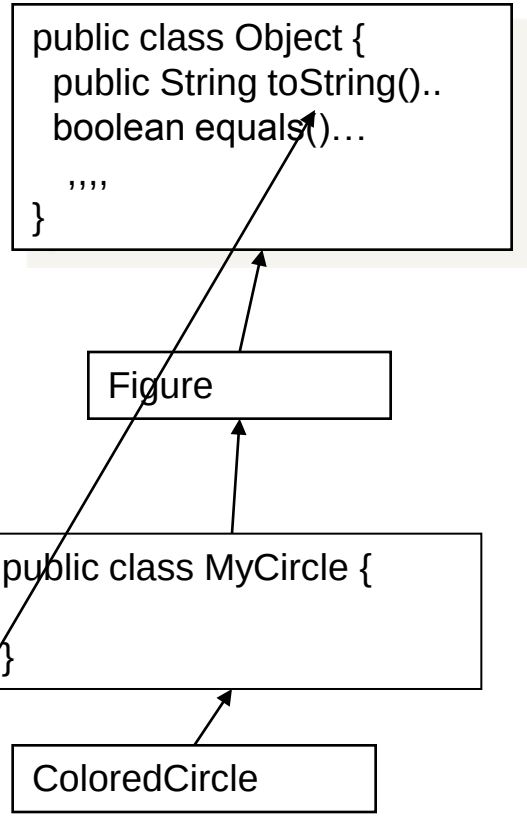


```
<終了> Ex21ToString [Java アプリケーション]
j2.lesson13.MyCircle@1b67f74
j2.lesson13.ColoredCircle@530daa
```

toString消去前

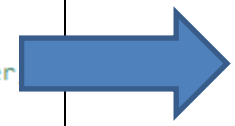


toString消去後




```
Figure fig = new MyCircle(50, 10, 100);

String s = fig.toString();
System.out.println(s);
```

```
15      /*
16      public String toString() {
17          String result = "[";
18          result += "x:" + this.x;
19          result += " y:" + this.y;
20          result += " diameter:" + this.diameter
21          result += "]";
22          return result;
23      }
24      */
```

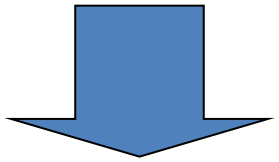


```
<終了> Ex21ToString [Java アプリケー
j2.lesson13.MyCircle@1b67f74
j2.lesson13.ColoredCircle@530daa
```

ObjectクラスのtoStringメソッドが実行されている。

ObjectクラスのtoStringメソッドを直接起動してみよ。

```
Object o = new Object();
String s3 = o.toString();
System.out.println(s3);
```



```
<終了> Ex21ToString [Java アプリケーション
java.lang.Object@ca0b6
```

■例題31

問題: 0で割る割り算を行ったときどのような例外が発生するか確認せよ。次に、上記の例外を捕捉(catch)することを確かめるプログラムを作成せよ。

（クラス名： Ex31ExceptionTest）

例題31(Ex31ExceptionTest)

```
1  package j2.lesson12;
2
3  public class Ex31ExceptionTest {
4      public static void main(String[] args) {
5          System.out.println(2 / 0);
6      }
7  }
```

```
<終了> Ex31ExceptionTest [Java アプリケーション] C:¥usr¥Java¥jre6¥bin¥javaw.exe (2011/04/01 19:35:38)
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at j2.lesson12.Ex31ExceptionTest.main(Ex31ExceptionTest.java:5)
```

```
1  package j2.lesson12;
2
3  public class Ex31ExceptionTest_1 {
4      public static void main(String[] args) {
5          try {
6              System.out.println(2 / 0);
7          } catch (ArithmeticException ex) {
8              System.out.println("0で割りました");
9          }
10     }
11 }
```

```
<終了> Ex31ExceptionTest_1 [Java アプリケーション] C:¥usr¥Java¥jre6¥bin¥javaw.ex
0で割りました
```

■例題32

問題: MyCircleに次のメソッドを作成し、動作を確認せよ。実行時例外（RuntimeException）を発生させるようにせよ。

返り値の型	メソッド名(引数)	機能
void	`testThrow()`	実行時例外（RuntimeException）を発生させる。

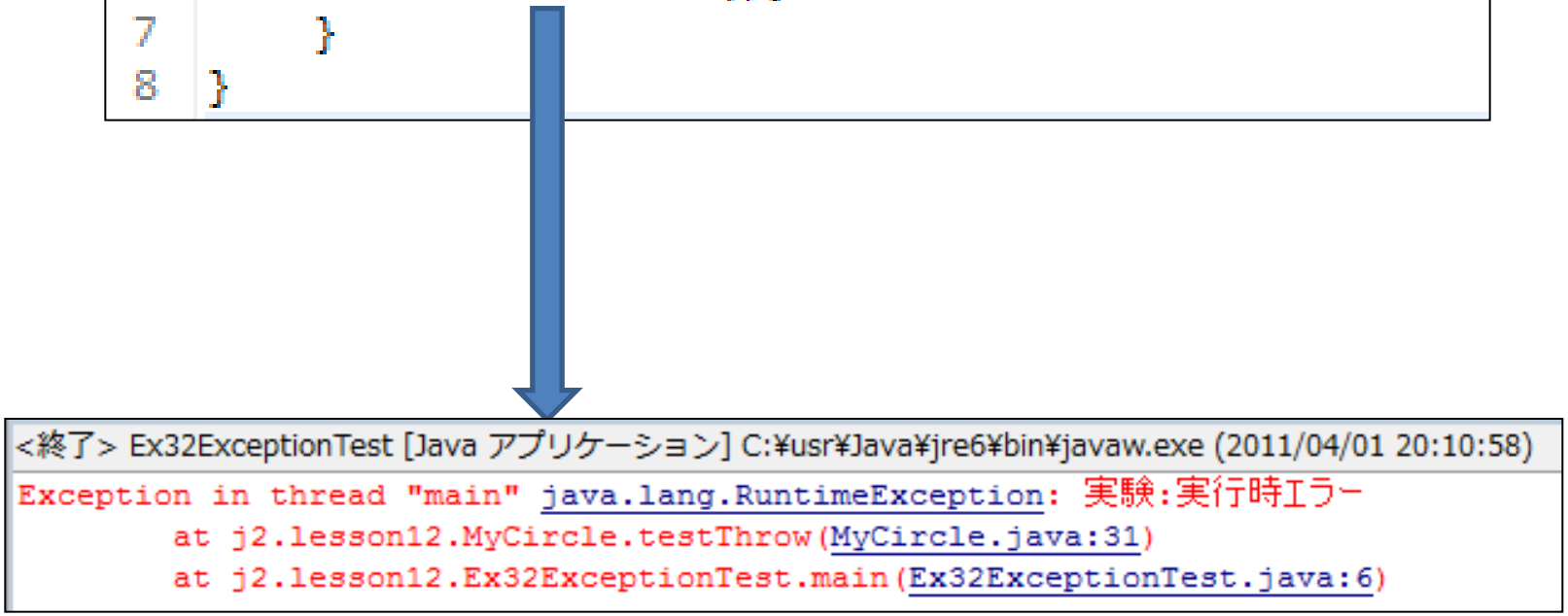
MyCircleを編集する
動作確認クラス：Ex32ExceptionTest

例題32(MyCircle)

```java
package j2.lesson12;

import gpjava.Canvas;

public class MyCircle extends Figure {
    int x;
    int y;
    int diameter;

    public  MyCircle(int x, int y, int d) {
        this.x = x;
        this.y = y;
        this.diameter = d;
    }

    public String toString() {
        String result = "[";
        result += "x:" + this.x;
        result += " y:" + this.y;
        result += " diameter:" + this.diameter;
        result += "]";
        return result;
    }

    public void draw() {
        Canvas.drawOval(this.x, this.y, this.diameter, this.diameter);
    }

    //Ex32
    public void testThrow() {
        throw new RuntimeException("実験:実行時エラー");
    }
}
```

例題32(Ex32ExceptionTest)

```
1  package j2.lesson12;
2
3  public class Ex32ExceptionTest {
4      public static void main(String[] args) {
5          MyCircle c = new MyCircle(50, 10, 50);
6          c.testThrow();
7      }
8  }
```



■例題41　オプション

(例題41は問題21-23の後に行う)

問題:次のインタフェースItemDataおよび抽象クラスItemListを利用してProductData, ProductListクラスを作成せよ。また、挙動を確認するためのプログラムEx41ProductListTestがどのように動作するか、考察せよ。

また、挙動を確認するためのプログラムEx41ProductListTestのmainメソッド内で呼ばれるメソッドがどのように動作するか考察せよ。(ItemData, ItemListクラスは、講義ページよりダウンロード可能)

(クラス名： ProductData,ProductList)
動作確認クラス:Ex41ProductListTest

インタフェースItemData

ItemData 抽象メソッド

返り値	メソッド名（引数）	説明
String	toString()	ItemDataの文字列表現を返す

クラスItemList

ItemListインスタンス変数

変数の型と名前	初期値	説明
String title	無し	表のタイトル
int index	0	配列のインデックス
int maxRecords	無し	配列のサイズ
ItemData[ ] contents	無し	ItemDataを格納する配列

（続く）

クラスItemList(続き)

ItemListインスタンスメソッド

返り値	メソッド名(引数)	説明
void	contents(int row)	Spreadsheet の　row 行目に、ItemListの内容を表示する
void	add(ItemData d)	引数のItemData型インスタンスを配列contentsに加える。
void	(abstract) addFromStrings(String [] strs)	(抽象メソッド)表の一列分を表すString配列を基に、データを一件作成し、それをリストに加える。
void	addItemsFromFile(String fileName)	引数のファイルで指定するファイルの内容からItemData型のインスタンスを生成し、配列に追加する。
void	addItemsFromFile()	ファイル選択ダイアログからファイルを選択し、その内容からItemData型のインスタンスを生成し、配列に追加する。
void	header(int row)	(抽象メソッド)Spreadsheetの row行に header を表示するための抽象メソッド。

ItemListコンストラクタ

コンストラクタ名(引数)	説明
ItemList(int maxRecords, String title)	登録できる要素数maxRecords個、表のタイトルtitleとして、表(リスト)を作成。

例題41(ItemData)

```java
package j2.lesson12;

public interface ItemData {
    public String toString();
    public void showItemData(int row);
}
```

例題41(ProductData)

```java
package j2.lesson12;

import gpjava.Spreadsheet;

public class ProductData implements ItemData {
    String name;
    int price;

    public ProductData(String name, int price) {
        this.name = name;
        this.price = price;
    }

    public String toString() {
        String result;
        result = "[商品名:" + name + ", 単価:" + price + "]";
        return result;
    }
    // showProductData() を showItemData に書き変える
    public void showItemData(int row) {
        Spreadsheet.setString(row, 0, name);
        Spreadsheet.setInt(row, 1, price);
    }

    public static void header(int row) {
        Spreadsheet.setString(row, 0, "商品名");
        Spreadsheet.setString(row, 1, "単価");
    }
}
```

showProductData() をインタフェースItemData に合わせて、showItemData() に書き変える

例題41 (ItemList)

```java
package j2.lesson12;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import javax.swing.JFileChooser;
import gpjava.Spreadsheet;

public abstract class ItemList {
    String title;
    int index = 0;
    int maxRecords;
    ItemData [] contents;

    public ItemList(int maxRecords, String title) {
        this.title = title;
        this.maxRecords = maxRecords;
        // this.contents = new ItemData [maxRecords];
    }

    public void contents(int row) {
        Spreadsheet.setString(row, 0, title);
        header(row + 1);
        for (int i = 0; i < index; i++) {
            ItemData p = contents[i];
            p.showItemData(row + i + 2);
        }
    }

    public ItemData add(ItemData p) {
        if (index == maxRecords) {
            return null; // これ以上、商品を登録できない。
        }
        contents[index] = p;
        index++;
        return p;
    }

```

例題41(ItemList) 続き

```
39      public abstract void addFromStrings(String [] strs);
40      public abstract void header(int row);
41
42      public void addItemsFromFile(String fileName) {
43          try {
44              BufferedReader reader = new BufferedReader(new FileReader(fileName));
45              int index = 0;
46              while (index < maxRecords) {
47                  String line = reader.readLine();
48                  if (line == null) { // null なら終了
49                      break;
50                  }
51                  String[] strs = line.split(",");
52                  this.addFromStrings(strs);
53                  index++;
54              }
55          } catch (IOException ex) {
56              ex.printStackTrace();
57              System.out.println("ファイルの読み込みに失敗しました。");
58          }
59      }
60
61      public void addItemsFromFile() {
62          JFileChooser ch = new JFileChooser() ;
63          int selected = ch.showOpenDialog(null);
64          if (selected == JFileChooser.APPROVE_OPTION) {
65              String fileName = ch.getSelectedFile().getPath();
66              this.addItemsFromFile(fileName);
67          }
68      }
69  }
```

addItemsFromFile(String)
から抽象メソッドaddFromStrings(String[])
が呼ばれることに注目

例題41(ProductList)

```
1  package j2.lesson12;
2
3  public class ProductList extends ItemList {
4      public ProductList(int maxRecords, String title) {
5          super(maxRecords, title);
6          this.contents = new ProductData[maxRecords];
7      }
8
9      public static ProductList createProductListFromFile(int maxRecords, String title) {
10         ProductList newObj = new ProductList(maxRecords, title);
11         newObj.addItemsFromFile();
12         return newObj;
13     }
14
15     public ProductData get(String aProduct) {
16         for(int i = 0; i < index; i++) {
17             ProductData p = (ProductData)contents[i];
18             if(p.name.equals(aProduct)) {
19                 return p;
20             }
21         }
22         return null;
23     }
24
25     public void addFromStrings(String[] strs) {
26         String name = strs[0];
27         int price = Integer.parseInt(strs[1]);
28         ProductData p = new ProductData(name, price);
29         this.add(p);
30     }
31
32     public void header(int row) {
33         ProductData.header(row);
34     }
35 }
```

例題41(クラス図)

```
public abstract class ItemList{
  String title;
  int index = 0;
  int maxRecords;
  ItemData [] contents;

  ItemList(int maxRecords, String title)

  void contents()
  ItemData add(ItemData p)
  abstract void addFromStrings(String [] strs);
  abstract void header(int row);
  void addItemsFromFile(String fileName)
  void addItemsFromFile()
}
```

```
public interface ItemData{
   public String toString();
}
```

```
public class ProductData implements ItemData {
  String name;
  int price;

  ProductData(String name, int price)

  String toString()
}
```

```
public class class ProductList extends ItemList {

  static WordList createProductListFromFile
     (int maxRecords, String title)

  ProductList(int maxRecords, String title)

  ProductData get(String aProduct)
  void addFromStrings(String[] strs)
  void header(int row)
}
```

```
public void addFromStrings(String[] strs) {
    String name = strs[0];
    int price = Integer.parseInt(strs[1]);
    ProductData p = new ProductData(name, price);
    this.add(p);

}
```

例題41(Ex41ProductListTest)

```java
package j2.lesson12;

import gpjava.Spreadsheet;

public class Ex41ProductListTest {
    public static void main(String[] args) {
        ProductList menu = new ProductList(10, "青果メニュー");
        menu.addItemsFromFile("inputFiles/12ProductData.txt");

        Spreadsheet.show();
        int row = 0;
        menu.contents(row);
        row += menu.index + 3;
        menu.get("apple").showItemData(row);
    }
}
```

inputFiles/12ProductData.txt

```
apple,100
grape,200
orange,300
strawberry,400
pineapple,500
melon,600
```

A	B	C
青果メニュー		
商品名	単価	
apple	100	
grape	200	
orange	300	
strawberry	400	
pineapple	500	
melon	600	
apple	100	