1 オブジェクトを作って動かす

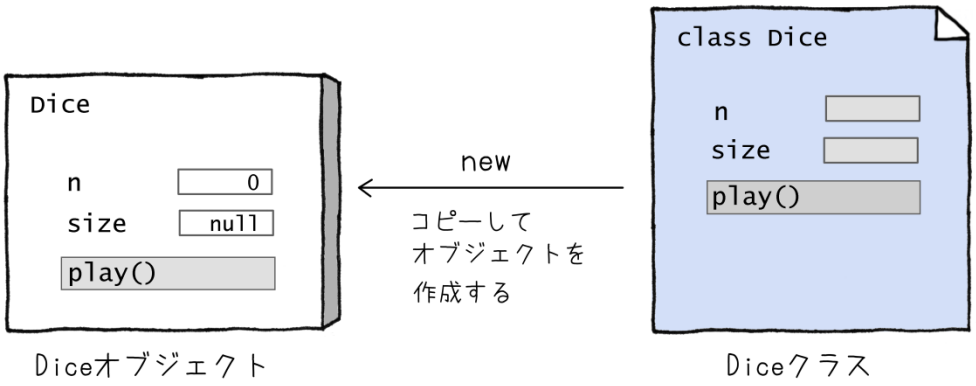
オブジェクト指向プログラミングでは、必要なオブジェクトをデザインして作成し、その機能を利用して目的の処理を実行します。この章では、オブジェクトの基本構成と作成方法から簡単な使い方までを解説します。オブジェクトの概念を確実に理解できるよう、工夫を凝らした多くのイメージ図も用意しました。オブジェクトとはどのようなもので、どうすれば使えるようになるか、例題と練習を通して体得してください。この章を理解すると、簡単なオブジェクトを作成して動かすことができるようになります。

内 容

1. クラスをデザインする
2. オブジェクトは new で作る
3. メソッドの中身を作る
4. オブジェクトから値を受け取る
5. オブジェクトに値を渡す

✧ まとめ

✧ 通過テスト

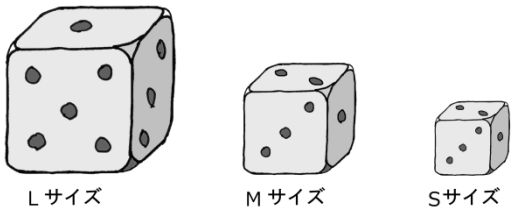


- クラスをコピーしてオブジェクトを作成する -

【プロジェクトとパッケージの作成についての注意】
①章ごとにプロジェクトを作成してください
　例えば、Part1 の 1 章では「Part1_1」という名前のプロジェクトを作成してください。
②各章のプロジェクト内には例題、練習、通過テストのパッケージを作成します
・例題 1 は sample1 パッケージ、例題 2 は sample2 パッケージのように作成します
・枝番号が付いた例題は同じパッケージに入れます。例えば、例題 1-1、例題 1-2 は共に sample1 パッケージに作成します。
・練習問題 1 は ex1 パッケージ、練習問題 2 は ex2 パッケージ、のように作成します
・通過テストは、問題 1 は pass1 パッケージ、問題 2 は pass2 パッケージ、のように作成します

1.クラスをデザインする

この章では、オブジェクト指向の流儀で、サイコロの機能や動作をプログラムします。Java はクラスという単位でプログラムを作るので、「サイコロクラス」を作ります。作成したサイコロクラスは、ひとつのプログラム部品になります。



サイコロクラスを作るため、サイコロの特徴を考えましょう。いろいろなことが考えられますが、とりあえず次の 3 つをあげてみました。

①サイコロは 1～6 のどれかの数字を値としてもつ ・・・・・ 目数(めかず)
②いろいろな大きさのサイコロがある・・・・・・・・・・・・・・・ サイズ
③振って転がすと新しい値に変えることができる ・・・・・・ 振って値を変更する

①と②はサイコロの属性です。「目数が 1 でサイズが"L"」のサイコロや「目数が 6 でサイズが"M"」のサイコロのように、属性の違いろいろなサイコロがあるので、目数やサイズを指定するための変数を使います。

また、③はサイコロの機能です。これはメソッドとして実現します。

例題 1-1　サイコロのクラス

Dice.java

```
1   package sample1;
2   public class Dice {
3       int       n;              // 目数
4       String    size;           // サイズ      } フィールド変数
5       void      play(){}        // サイコロを振る(中身は後で) --- メソッド
6   }
```

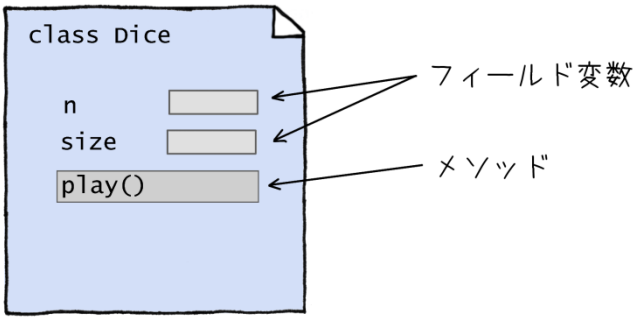
★sample1 パッケージに作成してください

【解説】

3、4 行で目数とサイズを記録するための 2 つの変数を宣言しています。変数をメソッドの外で宣言していることに注意してください。このようなメソッドの外で宣言した変数をフィールド変数といいます。フィールド変数は、クラスの中のすべてのメソッドからアクセスできます。

5 行目に、play メソッドを書いていますが、今はメソッドの形を作っただけです。処理内容は、もう少し後で書くことにします。メソッド名だけでもエラーにはなりません。

次は、作成した Dice クラスの模式図です。2 つのフィールド変数と 1 つのメソッドからなるクラスであることがわかります。このようなフィールドとメソッドを持つクラスは、典型的な Java のクラスです。



このようなクラスをどう使うのか、使い方はこの後で説明します。ここでは、サイコロの属性をクラスのフィールド変数とし、また、サイコロの機能をメソッドとしたことが理解できれば十分です。

練習 1

※プログラムは ex1 パッケージに作成しなさい

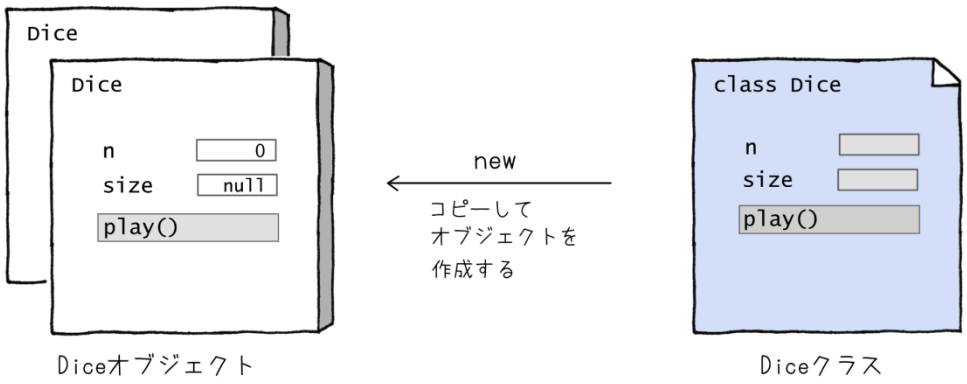
1.次の問に答えなさい。

問 1　本は①題名（title）、②著者（author）、③発行年（year）、という属性を持ちます。（例：　"伊豆の踊子"、"川端康成"、"1926"）」
機能は考えない事にします。この時、本のクラス（Book クラス）を作成しなさい（メソッドはありません）。

問 2　トランプのカードは、①種類（suit）、②数（number）、③表側が見えているかどうか（visible）、という属性を持ちます（例：　"ハート"、7、true）。「表かどうか」は boolean 型の変数を使います。表（true）、裏（false）です。
　また、ひっくり返す（表なら裏に、裏なら表にする、reverse）という機能を持ちます。reverse メソッドは、引数はなく、戻り値型は void です。
　このとき、トランプのカードのクラス（Card クラス）を作成しなさい。なお、メソッドは名前だけで、内容は作成しなくてもよい。

2.オブジェクトは new で作る

プログラムでサイコロが 2 つ以上必要な時、もうひとつ別な Dice クラスを作らねばならないのでしょうか？幸いなことに、その必要はありません。Java では、実際に動くプログラムは、作成した Dice クラスをコピーして、別に作るからです。



クラスをコピーして作り出した、実際に動かせるプログラムを**オブジェクト**といいます。図のように、フィールドだけでなくメソッドまで完全にコピーされます。また、フィールドには 0 や null などの初期値もセットされます。

そして、コピーは必要なだけいくつでも作成できます。プログラムの中で実際に仕事を行うのは、クラスではなくこれらのオブジェクトです。

オブジェクトを作成するには、new 演算子を使って次のように書きます。Dice の後ろに()を付けることに注意してください。

```
new Dice();
```

クラスは、それをコピーしてオブジェクトを作成するためのひな型です。そこで、Java 言語では、クラスをプログラマが作成した新しい型とみなします。ただし、int や double など**基本データ型**と区別するため、これを**クラス型**といいます。

そして、クラス型でも変数を宣言できるので、次のように、変数を宣言して、作成したオブジェクトを代入します。これが、オブジェクトを作成する一般的な方法です。

```
Dice d = new Dice();
```

なお、次の表に示すように、オブジェクトを作成すると、フィールド変数は既定値に初期化されます。これを**オブジェクト初期化の規定値**といいます。

規定値ではなく、任意の値を初期値として設定する方法は次の 2 章で解説します。

【初期化時の既定値】

データ型	既定値
boolean	false
char	'¥u0000' (==0)
byte	0
short	0
int	0
long	0
float	0.0
double	0.0
String	null
配列型	null

では次に、オブジェクトを作成する例を示します。

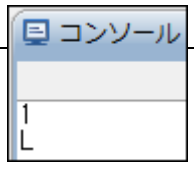
例題 1-2　オブジェクトを使う

Exec.java

```
1   package sample1;
2   public class Exec {
3       public static void main(String[] args) {
4           Dice  d      =   new Dice();    // オブジェクトを作成して d に代入
5           d.n          =   1;             // フィールド変数 n に 1 を代入
6           d.size       =   "L";           // 同じく size に"L"を代入
7           System.out.println(d.n);        // n を表示
8           System.out.println(d.size);     // size を表示
9       }
10  }
```

★Exec クラスは Dice クラスのある sample1 パッケージに作成してください。

実行結果 ▷



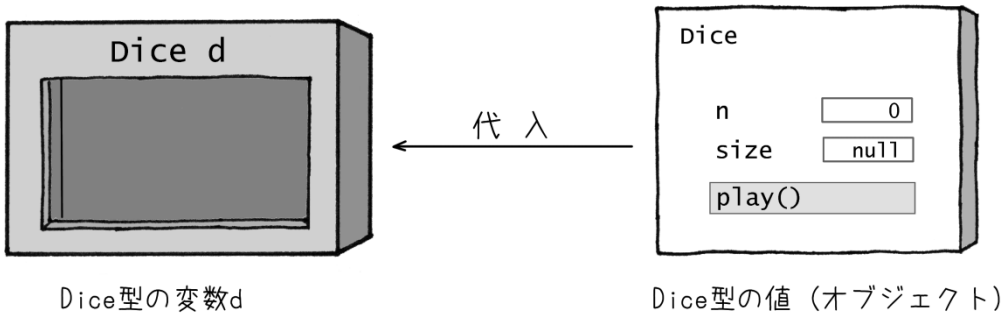
【解説】

Java 言語では、実行する処理は、main メソッドの中に書かねばなりません。オブジェクトの作成も例外ではなく、main メソッドの中で行います。したがって、オブジェクトを作成して、何かの仕事をするには、main メソッドを持つクラスが必要です。

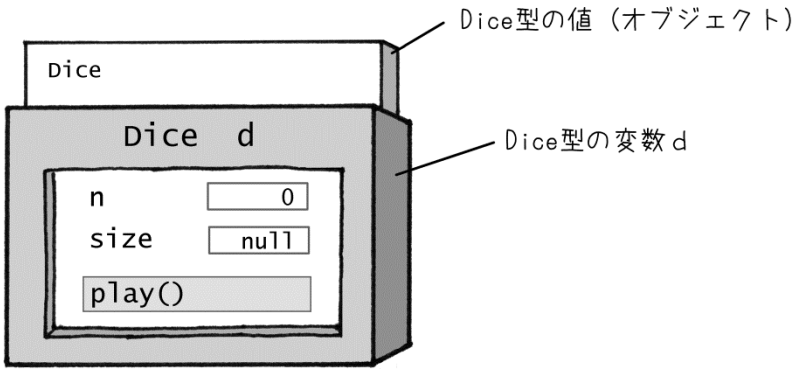
例題の Exec クラスは main メソッドだけを持つ、プログラム実行用のクラスです（Dice クラスを操作するので、Dice と同じ sample1 パッケージ内に作成します）。4 行目で、Dice 型の変数 d を宣言し、new で作成した Dice 型のオブジェクトを代入しています。フィールドと共に、メソッドもコピーされることに注意してください。

```
Dice d = new  Dice();  // Dice 型のオブジェクトを作成して d に代入する
```

次の図は、左側が Dice 型の変数、右側が Dice 型のオブジェクトです。宣言しただけの変数は"空"です。これは、変数 d に Dice 型のオブジェクトが代入される直前の様子です。



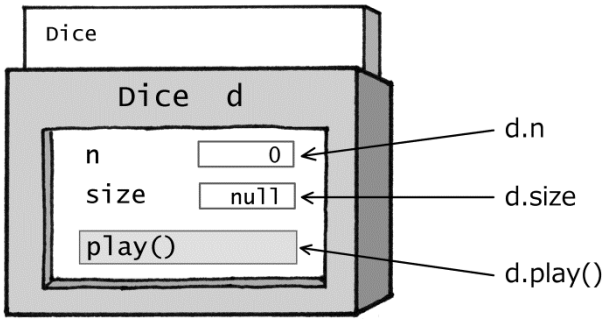
次は、代入後の変数の様子です。変数 d には Dice 型のオブジェクトが入っています。



■ 変数名.名前 の形式でフィールド変数やメソッドを指定する

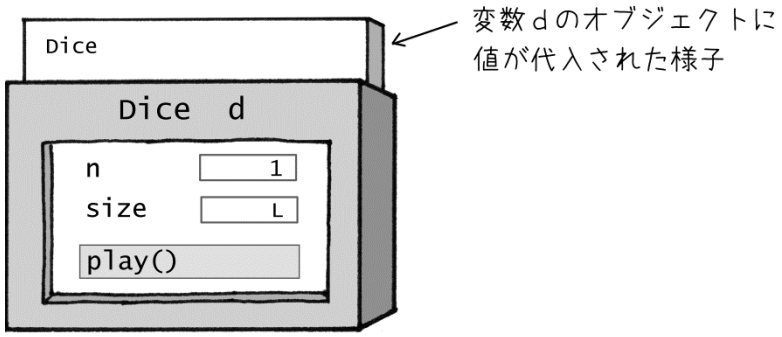
オブジェクトのフィールド変数に値を代入したり、あるいはセットされている値を参照したりできます。ただし、変数名とドットで連結して、d.n とか d.size のように使わねばなりません。

ドット（.）はメンバ参照演算子といいます。また、メンバとはオブジェクトの構成要素であるフィールド変数とメソッドのことです。例では n、size、play()がメンバです。



5～6 行目ではフィールド変数 n と size に値を代入しています。これがオブジェクトのメンバに値をセットするもっとも簡単な方法です。

```
d.n     =  1;          // フィールド変数 n に 1 を代入
d.size  =  "L";        // 同じく size に"L"を代入
```



また、7～8 行目では n と size の値を表示しています。

```
System.out.println(d.n);      // n を表示
System.out.println(d.size);   // size を表示
```

このように 変数名.名前 の形で使うのは、どのオブジェクトかを指定するためです。オブジェクトはいくつでも作れるため、オブジェクトが格納されている変数を指定する必要があることに注意してください。

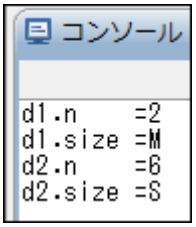
練習 2

※プログラムは ex2 パッケージに作成しなさい

1.Dice クラスのオブジェクトを 2 つ作成し、それぞれ変数 d1, d2 に代入します。その後、d1、d2 オブジェクトのフィールド変数に表の値をセットし、 実行結果のように表示しなさい。

変数	n	size
d1	2	"M"
d2	6	"S"

実行結果 ▷



＜注意＞

・sample1 パッケージの Dice クラスを ex2 パッケージにコピーしなさい

3.メソッドの中身を作る

オブジェクトの中にコピーされるplay()のようなメソッドを、インスタンスメソッドといいます。メソッドにはクラスメソッドとインスタンスメソッドがあります。一般に、Javaでメソッドというとインスタンスメソッドを指します。そのため、クラスメソッドの場合は必ず"クラス"を付けるようにします。

クラスメソッドについてはこの後でもう少し詳しく解説しますが、この節の主な内容はインスタンスメソッドの中身を作成することです。次の例題でその方法を解説します。

例題 2-1　メソッドの作成

Dice.java

```
1   package sample2;
2   public class Dice {
3       int       n;          // 目数
4       String    size;       // サイズ
5       void      play(){     // サイコロを振る
6           n = (int)(Math.random()*6) + 1; // 1～6 のどれかを n に代入する
7       }
8   }
```

【解説】

6 行目が新たに作成した play メソッドの中身です。サイコロの目数である n の値を、乱数を使ってセットします。n はフィールド変数であることに注意してください。このようにインスタンスメソッドの主な役割は、フィールド変数を操作することです。

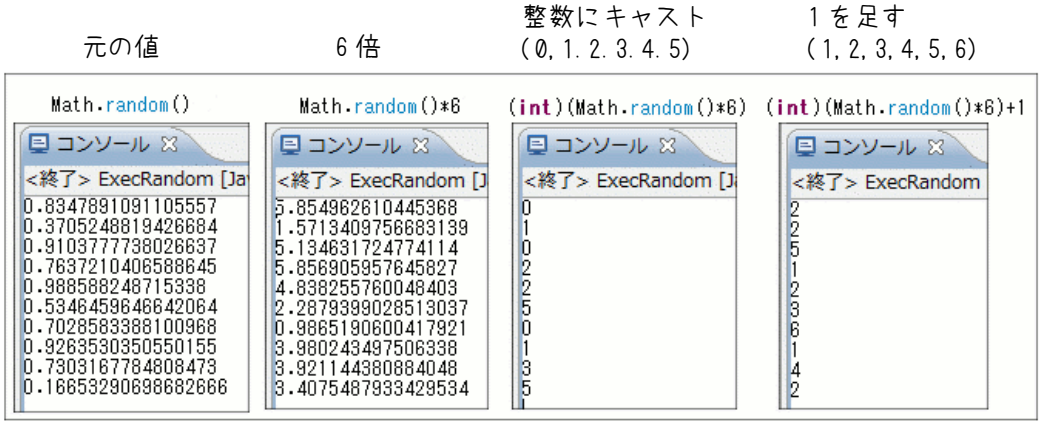
```
n = (int)(Math.random()*6) + 1; // 1～6 のどれかを n に代入する
```

Math.random()は標準クラスのメソッドで、0～1 の間の値を返します。値は実行するたびに違う値となり、予測できないので乱数といいます。偶然性の必要なゲームなどで利用されます。

```
0≦ Math.random()<1    --- Math.random()は 0 以上 1 未満の値です
```

例題では、この値を 6 倍したものを整数にキャストし、さらに 1 を足すことで 1,2,･･･6 の乱数に変換しています。このやり方は整数型の乱数を作る定石です。

次の図は、Math.random()を 10 回実行した時の値です。左端が random()の値、次がそれを 6 倍した値、さらにその次は、6 倍した値を int にキャストした値、最後の右端が、キャストしたものに 1 を足した値で、例題の 6 行目と同じです。1～6 の乱数が得られていることを確認してください。



（注）乱数なので 4 つの実行結果に表示される値はそれぞれ違います

次の例題は、このように作成したメソッドの使い方です。

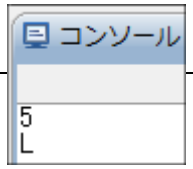
★sample2 パッケージに作成してください

例題 2-2　メソッドの使い方

Exec.java

```
1   package sample2;
2   public class Exec {
3       public static void main(String[] args) {
4           Dice  d   =   new Dice();        // オブジェクトを作成
5           d.n       =   1;                 // 目数を１に設定
6           d.size    =   "L";               // サイズを"L"に設定
7           d.play();         // サイコロを振って、設定した n の値が変わるか調べる
8           System.out.println(d.n);         // 表示
9           System.out.println(d.size);
10      }
11  }
```

実行結果 ▷



【解説】

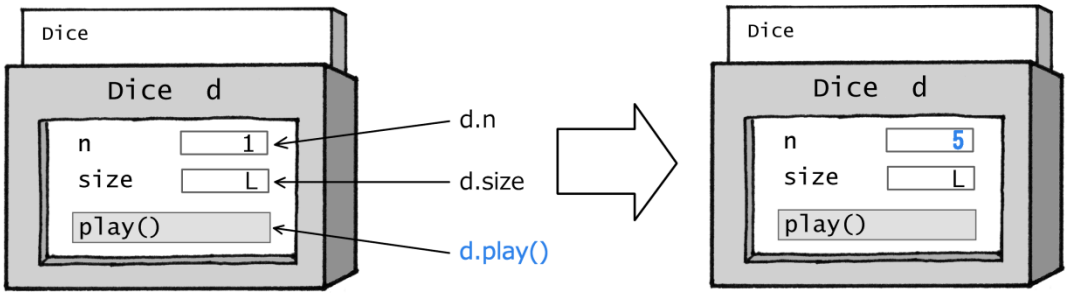
前の例題 1-2 でもそうでしたが、オブジェクトを作成して、何かの仕事をするには、main メソッドを持つクラスが必要です。この Exec クラスは、main メソッドだけを持つ実行用のクラスです。

4 行目で Dice オブジェクトを作成して変数 d に代入し、5、6 行目でフィールド変数 n を 1 に、また、size を"L"に初期化しています。ここまでは例題 1-2 と同じで、オブジェクトの作成とセットアップです。

次の 7 行目では、サイコロを振って n の値が変わるか調べます。play メソッドを実行して、5 行目で設定しておいた 1 が変更されることを確認しています。play メソッドは変数 d の中にある Dice オブジェクトのメンバなので、 d.play()と指定して呼び出します。

```
d.play();     // play メソッドで目数を変更
```

― 変数名 . メソッド の形で 使う ―



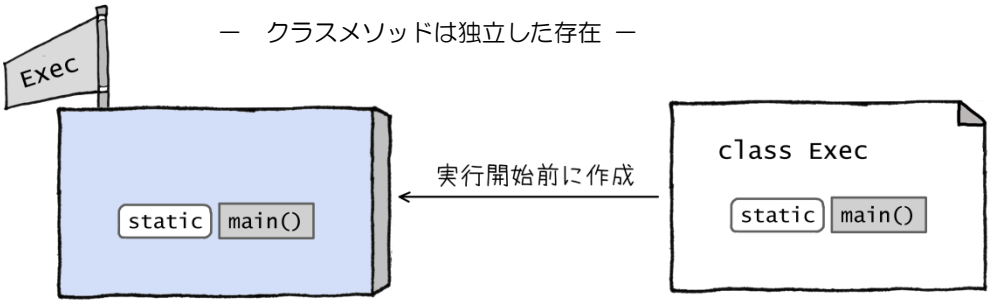
8、9 行で n と size の値を表示していますが、実行結果をみると、n は 5 となっており、play メソッドで値が変更されたことがわかります。何度か実行してみて、値がランダムに変わることを確認してください。

■ インスタンスメソッドとクラスメソッドの違い

main メソッドのように、static を付けて作成したメソッドをクラスメソッド（静的メソッド）といいます。クラスメソッドは、プログラムの実行開始直前にひとつだけ作成され、メモリ上の特定の場所に置かれます。

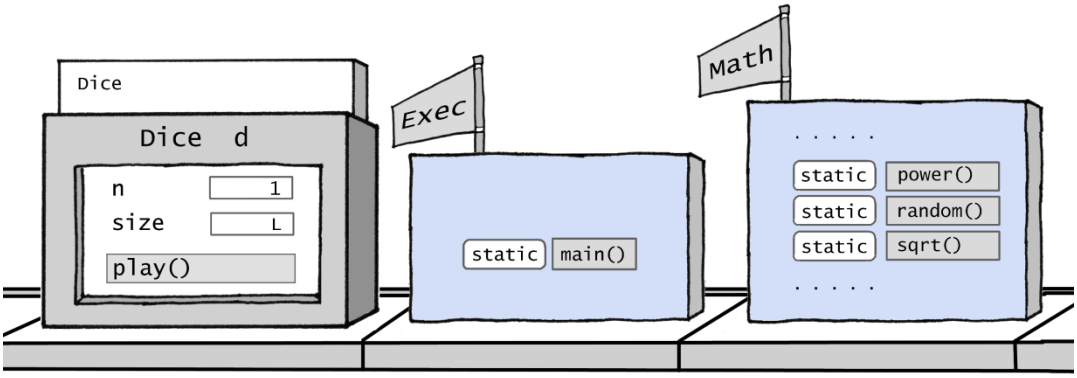
クラスから作成されますが、実体は常に 1 つだけしかありません。インスタンスメソッドのようにオブジェクトの中にコピーされず、メモリー上に単独で存在するメソッドです。

― クラスメソッドは独立した存在 ―



インスタンスメソッドは、new 演算子でオブジェクトの中にコピーされて初めて実体化しますが、クラスメソッドは実行開始前から常に存在します。そのため、いつでも呼び出して実行できます。

例えば、main メソッドはプログラムで最初に実行されるメソッドなので、クラスメソッドになっています。また、標準クラスの中にも、多くのクラスメソッドがあります。この例題で使った Math クラスのメソッドは、random を始め、どれもクラスメソッドです。



コンピュータのメモリー上に配置された Dice 型の変数 d とクラスメソッド
― クラスメソッドはオブジェクトには含まれない ―

図に示すように、クラスメソッドでは、クラス名の旗が、置かれた場所を示す目印として使われています。そこで、普通は、クラス名を指定して Math.random()のように呼び出さねばなりません。これがクラスメソッドと呼ぶ理由です。

クラスメソッドはクラス名を付けるだけで簡単に呼び出せるため、便利な機能を持つさまざまなクラスメソッドが、Java 言語のライブラリとして提供されています。

インスタンスメソッドとクラスメソッドの違い

	インスタンスメソッド	クラスメソッド
作成時期	プログラム実行中	実行開始前
作 成 数	いくつでも作成可能	1つだけ
存在形式	オブジェクトの中に存在	単体で存在
使用方法	変数名.メソッド名 例） d.play()	クラス名.メソッド名 例）Math.random()

練習 3

1.次の説明を読み、Counter クラスと、それをテストする Exec クラスを作成しなさい。なお、二つのクラスは、共に ex3_1 パッケージに作成しなさい。

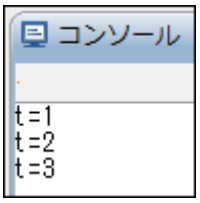
Counter クラス

何かの回数をカウントするクラスです。回数を表すフィールド変数 t を持ちます。また、init メソッドは t を 0 に初期化し、add メソッドは t の値をひとつ増やします。メソッドの戻り値型は共に void です。

フィールド変数	t	回数
メソッド	init()	t を 0 にする
	add()	t に 1 加える

Exec クラス

main メソッドの中で、Counter クラスのオブジェクト c を作成します。そして、init メソッドを実行して c のフィールド変数 t を 0 に初期化した後、add メソッドを 3 回呼び出します。また、呼び出す度に、t の値をコンソールに実行結果のように表示します。



△ 実行結果

2. 次の問に答えなさい。

問 1　練習 1 の問 2 で作成した次のような Card クラスの reverse メソッドを作成しなさい

```
public class Card {
    String     suit;          // カードの種類（スペードやハートなどの種類）
    int        number;        // カードの札番号
    boolean    visible;       // 表かどうか
    void       reverse(){}    // ひっくり返す
}
```

ただし、ex3_2 パッケージを作成し、その中に Card クラスをコピーしてから reverse メソッドの中身を作成しなさい。reverse メソッドは visible の値を反転するメソッドです。フィールド変数 visible が true なら false に、false なら true にします。

＜ヒント＞否定の!演算子を使って visible の値を反転すると簡単です

問 2　ex3_2 パッケージに Card クラスをテストする Exec クラスを作成しなさい

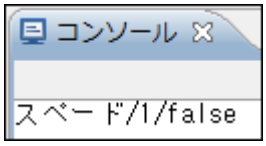
Exec クラスでは次のように処理を実行します。

①Card 型の変数 c を作成し、フィールド変数に次のような値をセットします

種類＝スペード、札番号＝1、表が見えている＝true

②reverse メソッドを実行する

③実行結果のように、c のフィールド変数の値を表示する



△ 実行結果

4.オブジェクトから値を受け取る

フィールド変数の値や計算結果など、オブジェクトから何かの値を受け取りたいことがあります。そのような場合は、クラスの中に return 文で値を返すメソッドを作成しておきます。

次の例は、フィールド変数の値を返すメソッドを作成しています。

例題 3　return 文で値を返すメソッド

Dice.java

```
1   package sample3;
2   public class Dice {
3       int       n;                  // 目数
4       String    size;               // サイズ
5       void      play(){             // サイコロを振る
6           n = (int)(Math.random()*6) + 1;
7       }
8       int       getN()     { return   n;     }   // 目数を返す
9       String    getSize()  { return   size;  }   // サイズを返す
10  }
```

Exec.java

```
1   package sample3;
2   public class Exec {
3       public static void main(String[] args) {
4           Dice  d   =   new Dice();
5           d.n       =   6;          // フィールド変数 n に 6 をセットする
6           d.size    =   "M";        // フィール変数 size に"M"をセットする
7           System.out.println(d.getN());      // n の値を取得して、表示
8           System.out.println(d.getSize());   // size の値を取得して、表示
9       }
10  }
```

実行結果 ▷

【解説】

例題は、Dice クラスと、それをテストする Exec クラスです。Dice クラスは、見やすくするために、8、9 行目のメソッドを、それぞれ 1 行で書いています。

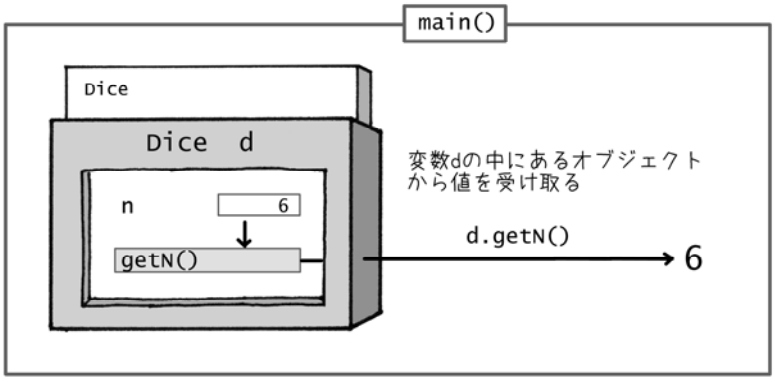
8 行目の getN()は、return 文でフィールド変数 n の値を返します。また、9 行目の getSize()は、フィールド変数 size の値を返します。つまり、getN()はサイコロの目数を返し、getSize()はサイコロのサイズを返します。

返す値に応じて、戻り値型が int と String になっていることにも注意して下さい。

```
int      getN()      { return  n;    }  // 目数を返す
String   getSize()   { return  size; }  // サイズを返す
```

（注）getN()や getSize()のように、2 つ以上の単語をつないでメソッド名を作るときは、後ろに連結する単語の先頭を大文字にするのが一般的です。ラクダのこぶのように途中に大文字が混ざるので、これをキャメルケースといいます。

オブジェクト内のメソッドが、このように return 文で値を返すことにより、オブジェクトはその外側の世界に値を渡すことができます。



Dice オブジェクトをテストする、Exec クラスの main メソッドでは、d.getN()や d.getSize()を使用して、変数の中にあるオブジェクトから値を受け取り、表示しているわけです。

このようにオブジェクトのメソッドを使ってフィールド変数の値を得るのは、d.n のように、オブジェクトのフィールド変数を直接アクセスするよりもよい方法です（その理由は、次の章で説明します）。

```
System.out.println(d.getN());       // n の値を取得して、表示
System.out.println(d.getSize());    // size の値を取得して、表示
```

練習 4

※プログラムは ex4 パッケージに作成しなさい

1.次の問に答えなさい。

問 1 練習 3-2 で作成した Card クラスに、以下のメソッドを追加しなさい

(1)カードの種類を返す getSuit メソッド

(2)カードの札番号を返す getNumber メソッド

(3)表が見えているかどうかを返す isVisible メソッド

※Jave 言語ではフィールド変数が boolean 型である場合、その値を返すメソッドは、get ではなく is を前に付けます。ですから、isVisible はフィールド変数 visible の値を返すメソッドです。

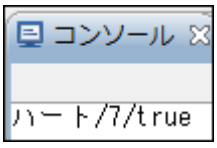
問 2 Card クラスをテストする Exec クラスを作成し、main メソッドに次のような処理を作成しなさい。

①Card クラスのオブジェクトを作成し、Card 型の変数 c に代入する

②種類＝"ハート"、数＝7、表が見えている＝true、と言う値を持つように、c のフィールド変数を設定する

③getSuit、getNumber、isVisible を使って、作成したカードの内容を実行結果のように表示する

実行結果 ▷



5.オブジェクトに値を渡す

オブジェクトに何かの値を渡して、処理を行わせたいことがあります。そのような場合は、クラスの中に引数を持つメソッドを作成しておきます。これによりメソッドの引数としてオブジェクトに値を渡すことができます。

次は、メソッドに渡した引数でオブジェクトのフィールド変数を書き変える例です。

例題 4　引数を持つメソッド

Dice.java

```
1   package sample4;
2   public class Dice {
3       int       n;                  // 目数
4       String    size;               // サイズ
5       void  play(){                 // サイコロを振る
6           n = (int)(Math.random()*6) + 1;
7       }
8       int       getN()         { return   n;      }   // 目数を返す
9       String    getSize()      { return   size;   }   // サイズを返す
10      void  setN(int m)        { n      =  m;  }       // 目数をセット
11      void  setSize(String s) { size =   s;  }       // サイズをセット
12  }
```

Exec.java

```
1   package sample4;
2   public class Exec {
3       public static void main(String[] args) {
4           Dice  d   =   new Dice();
5           d.setN(6);           // フィールド変数 n に 6 をセットする
6           d.setSize("M");      // フィール変数 size に"M"をセットする
7           System.out.println(d.getN());      // n の値を取得して、表示
8           System.out.println(d.getSize());   // size の値を取得して、表示
9       }
10  }
```

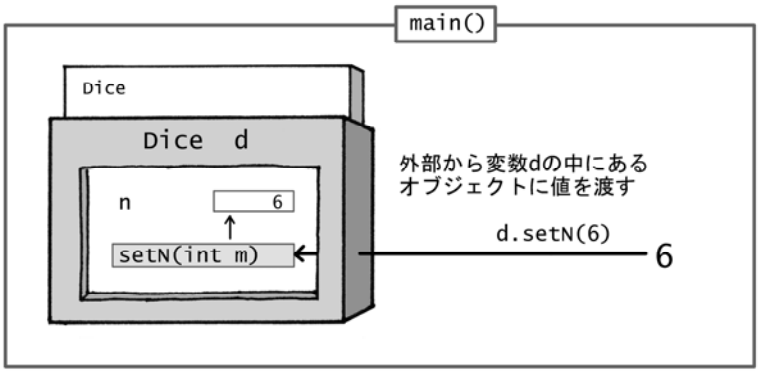
実行結果 ▷

【解説】

例題は、Dice クラスとそれをテストする Exec です。Dice クラスの 10 行目にある setN メソッドは、引数 m をフィールド変数 n に代入するメソッドです。また、11 行目の setSize メソッドも、引数 s をフィールド変数 size に代入するメソッドです。

```
void  setN(int m)          { n    = m; }  // 目数をセット
void  setSize(String s)    { size = s; }  // サイズをセット
```

これらのメソッドの引数に値をセットして呼び出すことにより、外部からオブジェクトへ値を渡して、フィールド変数の値を変更することができます。



Exec クラスの main メソッドでは、Dice オブジェクトを作成し、setN メソッドと setSize メソッドを使って、オブジェクトのフィールド変数に値をセットしています。

このように、オブジェクトのメソッドを使ってフィールド変数の値を変更するのは、d.n のように、オブジェクトのフィールド変数を直接アクセスするよりも良い方法です（その理由は次の章で説明します）。

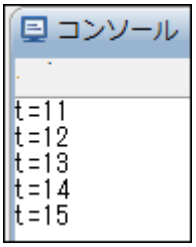
```
d.setN(6);          // フィールド変数 n に 6 をセットする
d.setSize("M");     // フィール変数 size に"M"をセットする
```

練習 5

※プログラムは ex5 パッケージに作成しなさい

1.練習 3-1 で作成した Counter クラスに、フィールド変数の t の値を返す getT メソッドと、フィールド変数 t に値をセットする setT メソッドを追加しなさい。また、Counter クラスをテストする Exec クラスを作成し、次のような処理を行いなさい。

①Counter クラスのオブジェクトを作成し、Counter 型の変数 c に代入する
②作成したオブジェクトのカウンターを 10 にセットする
③add メソッドを 5 回実行し、実行結果のように、毎回カウンタの値を表示する（for 文で）



△ 実行結果

オブジェクトとインスタンスの違い

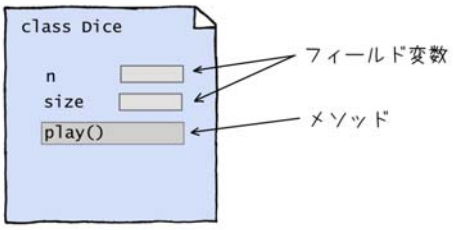
オブジェクトとインスタンス（instance、例という意味）はほぼ同じような意味で使われています。例えば、Dice d = new Dice(); としたとき、「d は Dice 型のオブジェクトである」という一方で、「d は Dice クラスのインスタンスである」ともいいます。オブジェクトは"型"を意識したときに使い、インスタンスは"クラス"を意識したときに使う、それだけの違いです。

オブジェクトとインスタンスを使い分けると、Java に詳しくなったような気がしますが、クラスとオブジェクトの関係をしっかり理解することの方が大切です。

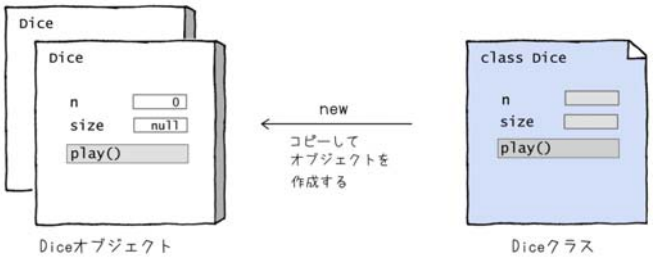
まとめ

クラスをコピーしてオブジェクトを作る

□クラスは、属性を表すフィールド変数と機能を実現するメソッドからなる



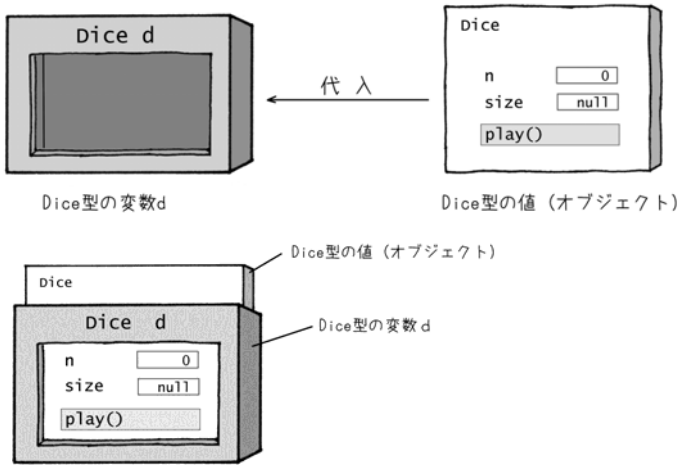
□クラスは、それをコピーしてオブジェクトを作成するためのひな型である



□クラスはプログラマが作成した新しい型とみなされる

□`int` や `double` などを基本データ型というのに対して、クラス型という

□オブジェクト作成は、`new` 演算子でクラスのコピーを作成してクラス型の変数に代入する



□同じクラスから、複数のオブジェクトをいくつでも作成できる

□オブジェクト作成時に、フィールド変数は初期化の規定値で初期化される

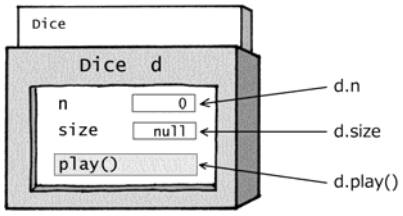
【初期化時の既定値】

データ型	既定値
boolean	false
char	'¥u0000' (==0)
byte	0
short	0
int	0
long	0
float	0.0
double	0.0
String	null
配列型	null

オブジェクトの中身

□オブジェクト内のフィールド変数やメソッドをメンバという

□変数に格納されたオブジェクトのメンバは、変数名.メンバ名と書き表す



□メソッドにはインスタンスメソッドとクラスメソッドがある

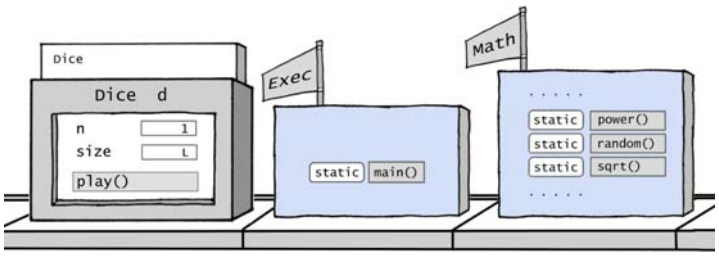
	インスタンスメソッド	クラスメソッド
作成時期	プログラム実行中	実行開始前
作 成 数	いくつでも作成可能	1つだけ
存在形式	オブジェクトの中に存在	単体で存在
使用方法	変数名.メソッド名 例) d.play()	クラス名.メソッド名 例) Math.random()

□変数に格納されたオブジェクトの中にあるメソッドを、インスタンスメソッドという

□インスタンスメソッドはオブジェクトの中で動く

□`static` キーワードの付いたメソッドは、クラスメソッドという

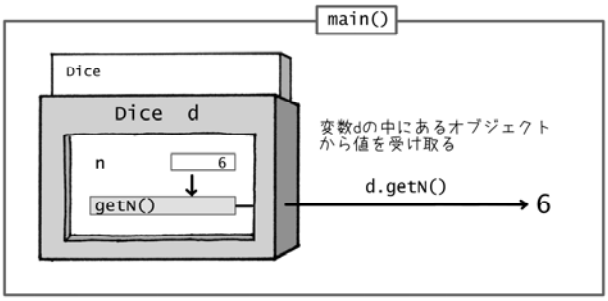
□クラスメソッドはプログラムの実行開始前に作成され、メモリーの特定の場所に置かれる



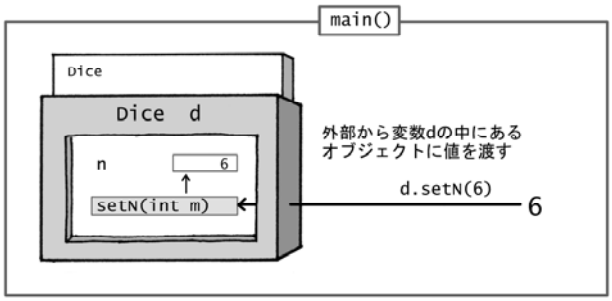
□クラスメソッドは常に存在するので、呼び出すだけで使用できる
□クラスメソッドは**クラス名.メソッド名**という表現で利用する

オブジェクトとの値のやり取り

□オブジェクトから値を受け取るには、オブジェクトのクラスに値を返すメソッドを作成する



□オブジェクトに値を渡すには、オブジェクトのクラスに引数を持つメソッドを作成する



通過テスト

1.空欄にあてはまる言葉をそれぞれの選択肢から選んで記号で答えなさい

・クラスはオブジェクトを作成するためのひな型で、プログラマが定義した新しい型とみなされる。int,double などを基本データ型というのに対して、これを（1）という。

・クラスを new 演算子でコピーしたものがオブジェクトである。（2）メソッドをインスタンスメソッドといい、Java でメソッドというと、普通はインスタンスメソッドのことである。

・static のついたメソッドをクラスメソッドという。クラスメソッドはプログラム実行開始前に一度だけ作成されるが、実行開始後、（3）。

・インスタンスメソッドはオブジェクトが入っている変数の名前にメンバ参照演算子を付けて呼び出すが、クラスメソッドは（4）にメンバ参照演算子を付けて呼び出す。

（1）～（4）に対する選択肢

（1） ア オブジェクト型　　イ クラス型　　ウ メンバ型

（2） ア クラスの中で動く　　イ オブジェクトの中で動く　　ウ フィールドで動く

（3） ア いつでも作成できる　　イ 新たに作成されることはない　　ウ コピーできる

（4） ア クラス名　　イ オブジェクト名　　ウ フィールド名　　エ メソッド名

【解答】（1）＿＿＿＿ （2）＿＿＿＿ （3）＿＿＿＿ （4）＿＿＿＿

2.例題 4 の Dice クラスにならって、目数が 1～52 であるようなサイコロ（正 52 面体のサイコロ）のクラス（Dice52）を作成しなさい.クラスのフィールドとメソッドは次の通り。

（※pass2 パッケージに作成しなさい）

＜Dice52 クラス＞

フィールド	int n	サイコロの目数
メソッド	int getN()	目数 n の値を返す
	void setN(int m)	目数 n に m をセットする
	int play()	サイコロを振って目数 n を変更する。また n を返す

3.次の問に答えなさい

（※pass3 パッケージに作成しなさい）

問１　次のようなトランプのカードクラスにメソッドを追加して完成させなさい。

```
public class Card {
     int        suit;      // 種類（0=スペード、1=ハート、2=クラブ、3=ダイヤ）
     int        number;    // 札番号（1～13）
     int        getSuit()   { return  suit; }        // 種類を返す
     int        getNumber() { return number; }       // 札番号を返す
}
```

※suit が int 型であることに注意

追加するメソッド

void setSuit(int s)	フィールド変数 suit に引数 s をセットする
void setNumber(int m)	フィールド変数 number に引数 m をセットする
int toSuit(int n)	引数 n（1～52 の通し番号）から、カード種類の値を計算して返す
int toNumber(int n)	引数 n（1～52 の通し番号）から、札番号を計算して返す

＜説明＞

トランプのカードは 52 枚（4 種類×13 枚）あるので、1～52 の通し番号で処理すると、操作が簡単になります。しかし、そのためには、通し番号からカード種類の値（0、1、2、3）や札番号（1～13）を計算するメソッドが必要です。

通し番号を n とすると、n からカードの種類と札番号を求めるには次の式を使います。

種類 = (n-1)/13　------ 整数の割り算なので、0,1,2,3 のどれかになる

種類（0=スペード、1=ハート、2=クラブ、3=ダイヤ）

札番号= (n-1)%13+1　--- 1～13 のどれかになる

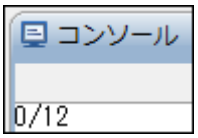
（計算例）通し番号 36 はクラブの 10

(36-1)/13 = 2 （クラブ）

(36-1)%13 + 1 = 10

toSuit メソッドは、通し番号を引数として受け取り、カード種類の値を計算して返します。また、toNumber メソッドは、通し番号を引数として受け取り、札番号を計算して返します。このような変換のための計算は Card 型オブジェクトと関係が深いので、Card クラスにメソッドとして加えることに、妥当性があると考えられます。

問 2　問 1 で作成した Card クラスのオブジェクトを 1 つ作成し、カード種類と札番号をセットして、実行結果のように表示しなさい。



△ 実行結果

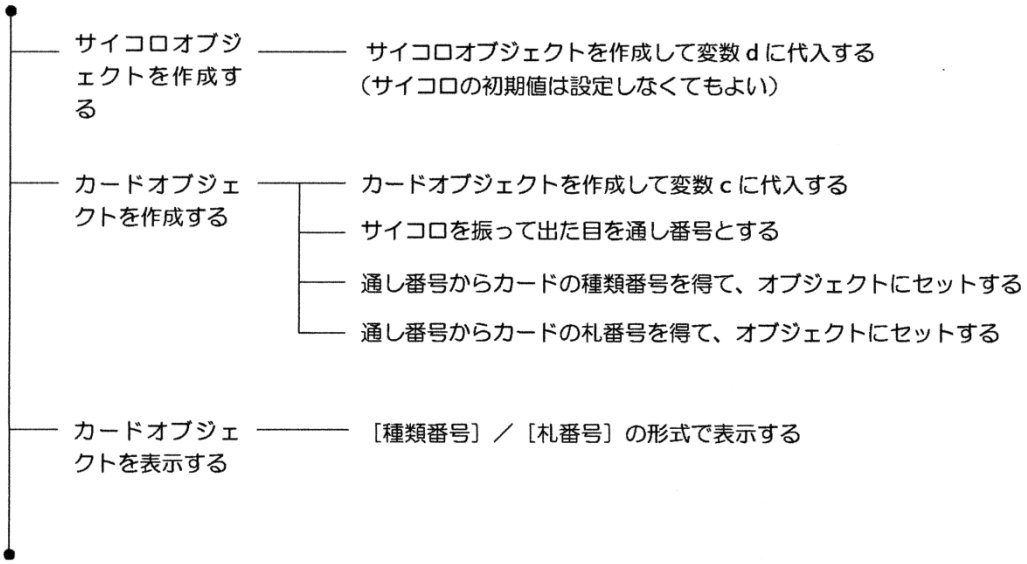
ただし、カード種類と札番号は、2.で作成した Dice52 クラスのオブジェクトを使って、1～52 の乱数を取得し、その値から計算で求めます。

手順は次の SPD（プログラム構造図）を参考にしなさい。

なお、Dice52 クラスが必要ですから、pass2 パッケージの Dice52 クラスをこのクラスを作成する pass3 パッケージに次の手順でコピーしなさい。

①pass2 パッケージの Dice52.java をマウスの右ボタンでクリックし［コピー］を選ぶ

②pass3 パッケージをマウスの右ボタンでクリックし［貼り付け］を選ぶ



通し番号から種類と札番号を計算するには、Card オブジェクトの toSuit メソッドと toNumber メソッドを使います。また、種類と札番号を、カードオブジェクトのフィールド変数にセットするには、setSuit メソッドと setNumber メソッドを使いなさい。

実行結果のように、種類／札番号という形式でカードオブジェクトのフィールド変数を表示しますが、フィールド変数の値を得るには、getSuit メソッドと getNumber メソッドを使いなさい。