

2023 年 12 月 7 日 (木) 実施

スーパークラスとサブクラス – 継承とオーバーライド –

あるクラスを拡張して作られたクラスを元のクラスの**サブクラス**と呼び、**サブクラス**から見て元のクラスを**スーパークラス**と呼ぶ。**あるクラスから複数のサブクラスを作ることは可能である**が、**あるサブクラスを複数のスーパークラスから作ることは出来ない**。(下図)



サブクラスの作り方は次の様になる。

```

修飾子 class サブクラス名 extends スーパークラス名 {
    型 フィールド名;
    修飾子 コンストラクタ名(=サブクラス名)( 引数 ) {
        コンストラクタの定義
    }
    修飾子 戻り値のデータ型 メソッド名( 引数 ) {
        メソッドの定義
    }
}
    
```

**サブクラスはスーパークラスのフィールド及びメソッドを引き継ぎ**、このことを**継承**と呼ぶ。**サブクラス**には、**スーパークラス**には無いフィールド及びメソッドを新規に定義出来ると共に、**スーパークラス**から継承したフィールド及びメソッドを再定義することも可能である。**サブクラスでスーパークラスから継承したメソッドを再定義することをオーバーライド(override)**と呼ぶ。なお、**サブクラスにはスーパークラスからコンストラクタは継承されない**ので注意が必要である。

Grade クラス・Grade2 クラス・Grade3 クラス

Eclipse で jimbo の様に『**自分の姓**』のパッケージを指定して、次の **Grade** クラス、**Grade2** クラス及び **Grade3** クラスを作成せよ。**(これは直接実行出来ないクラス ⇒ 後の例題に利用する)**

```

package jimbo;

class Grade {
    int point;
    char grade;

    Grade() {
    
```

```

        point = -1;
        setGrade(point);
    }

    Grade(int p) {
        point = p;
        setGrade(point);
    }

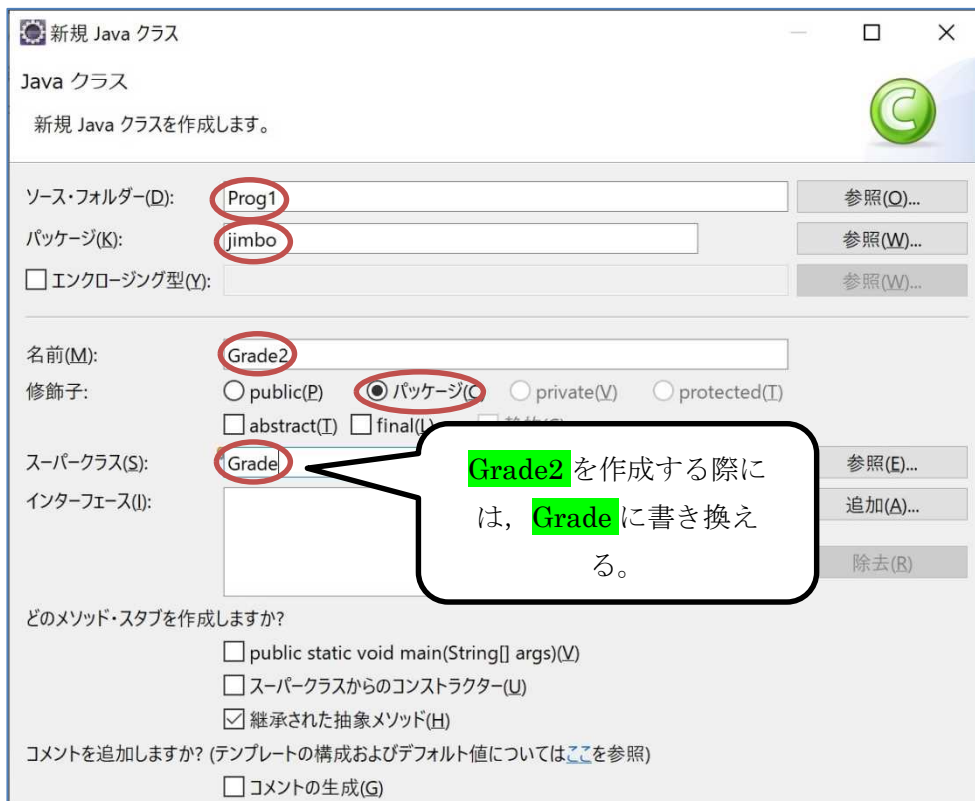
    void setGrade(int p) {
        int rank;

        if (p < 0 || p > 100)
            grade = 'X';
        else {
            rank = p/10;

            switch (rank) {
                case 6: grade = 'C'; break;
                case 7: grade = 'B'; break;
                case 8: grade = 'A'; break;
                case 9:
                case 10: grade = 'S'; break;
                default: grade = 'D';
            }
        }
    }
}

```

なお, eclipse では新規のクラスを作成する際にスーパークラスを指定することが出来る。



```

package jimbo;

class Grade2 extends Grade {
    Grade2() {
        this(-1);
    }

    Grade2(int p) {
        super(p);
    }

    void dispGrade() {
        System.out.println("点数: " + point + "--> 成績: " + grade);
    }
}

```

## 【解説】

1. `Grade2` クラスのコンストラクタで引数無しのものでは、`this(-1)` を呼び出している。この `this` はこのクラスのコンストラクタを表し、その中の引数を 1 つ取るものを呼び出す。
2. `Grade2` クラスのコンストラクタで引数を 1 つ取るものでは、`super(p)` を呼び出している。この `super` はスーパークラスのコンストラクタを表し、その中の引数を 1 つ取るものを呼び出す。

```

package jimbo;

class Grade3 extends Grade2 {
    Grade3() {
        this(-1);
    }

    Grade3(int p) {
        setPoint(p);
    }

    void setPoint(int p) {
        point = p;
        setGrade(point);
    }
}

```

## 例題 1

次のプログラムは `Grade2` クラスのインスタンスを複数作成し、`Grade2` クラスのコンストラクタやメソッドの働きを確認するものである。これを入力、ビルドして、実行せよ。ここで、クラス名は `Sample8_1`、ソースファイル名は `Sample8_1.java` とする。

```

package jimbo;

```

```
public class Sample8_1 {  
  
    public static void main(String[] args) {  
        // TODO 自動生成されたメソッド・スタブ  
        int po;  
  
        Grade2 gr2a = new Grade2();  
        Grade2 gr2b = new Grade2(85);  
  
        gr2a.dispGrade();  
        gr2b.dispGrade();  
  
        gr2a.setGrade(90);  
        gr2b.setGrade(78);  
  
        System.out.println("-----");  
  
        gr2a.dispGrade();  
        gr2b.dispGrade();  
  
        po = 65;  
        gr2a.point = po;  
        gr2a.setGrade(po);  
  
        po = 54;  
        gr2b.point = po;  
        gr2b.setGrade(po);  
  
        System.out.println("-----");  
  
        gr2a.dispGrade();  
        gr2b.dispGrade();  
    }  
}
```

## 例題 2

次のプログラムは `Grade3` クラスのインスタンスの配列を作成し、`Grade3` クラスのコンストラクタやメソッドの働きを確認するものである。これを入力、ビルドして、実行せよ。ここで、クラス名は `Sample8_2`、ソースファイル名は `Sample8_2.java` とする。

```
package jimbo;  
  
import java.util.Scanner;  
  
public class Sample8_2 {  
  
    public static void main(String[] args) {  
        // TODO 自動生成されたメソッド・スタブ  
        final int NUM = 3;  
        int[] po = new int[NUM];  
        Scanner sc = new Scanner(System.in);  
    }  
}
```

```

Grade3[] grs = new Grade3[NUM];

for (int i=0; i<NUM; i++)
    grs[i] = new Grade3();

System.out.println("初期値:");
for (int i=0; i<NUM; i++) {
    System.out.print((i+1)+"人目の ");
    grs[i].dispGrade();
}

System.out.println("-----");
for (int i=0; i<NUM; i++) {
    System.out.println((i+1)+"人目の");
    System.out.print("点数を入力してください:");
    po[i] = sc.nextInt();
    grs[i].setPoint(po[i]);
}

System.out.println("-----");
for (int i=0; i<NUM; i++) {
    System.out.print((i+1)+"人目の ");
    grs[i].dispGrade();
}
}
}

```

### 演習

次の `Grade4` クラス（ソースプログラム名は `Grade4.java`）の空欄 1) \_\_\_\_\_ 及び 2) \_\_\_\_\_ に適切な語句を埋めて保管した上で（これは直接実行出来ない）、`Ex8` クラスを入力、ビルドして実行せよ。なお、`Grade4` クラスは、`Grade3` クラスを継承して氏名の入力に対応出来る様にしたものである。

```

package jimbo;

class Grade4 extends Grade3 {
    String name;

    Grade4() {
        this(-1);
    }

    Grade4(int p) {
        this("None", p);
    }

    Grade4(String n, int p) {
        super( 1 );
        setName(n);
    }
}

```

```
void setName(String n) {
    2) _____ = n;
}

void dispGrade() {
    System.out.println("氏名: " + name + " ¥t 点数: " + point + "--> 成績: " +
grade);
}
}
```

**【解説】**

1. メソッドのオーバーライドは、メソッド名及び引数の数とデータ型とがスーパークラスと同一の場合に行われる。`Grade4` クラスでは `dispGrade` メソッドがオーバーライドされる。
2. スーパークラスのコンストラクタ呼び出し `super` は `Grade4` クラスのコンストラクタ内の最初の文として記述する必要がある。

```
package jimbo;

import java.util.Scanner;

public class Ex8 {

    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        final int NUM = 3;
        int[] po = new int[NUM];
        String[] na = new String[NUM];
        Scanner sc = new Scanner(System.in);

        Grade4[] grs = new Grade4[NUM];

        for (int i=0; i<NUM; i++) {
            System.out.println((i+1)+"人目の");
            System.out.print(" 氏名を入力してください: ");
            na[i] = sc.next();
            System.out.print(" 点数を入力してください: ");
            po[i] = sc.nextInt();
        }

        for (int i=0; i<NUM; i++)
            grs[i] = new Grade4(na[i], po[i]);

        System.out.println("-----");
        for (int i=0; i<NUM; i++){
            System.out.print((i+1)+"人目の ");
            grs[i].dispGrade();
        }
    }
}
```

提出物：

- 1) 例題 1, 例題 2 及び演習のプログラムのコンソールへの出力結果をコピーして貼り付けたテキストファイル `res8.txt` をメールに添付する。
  - 2) 演習で空欄を埋めたソースプログラムのファイル `Grade4.java` をメールに添付する。
  - 3) 第 8 回の理解度確認用の質問ファイル `Prog1_Questions_8th.txt` に解答を記入して, メールに添付する。
- \* メールのは名は『プログラミング 1 第 8 回課題』(鍵括弧は要らない) とする。