1 継承

1．予約語 extends

```
アクセスレベル class サブクラス名 extends スーパクラス名 {
   ...
}
```

(1) スーパクラス（既存のクラス）を拡張して，サブクラス（新しいクラス）を定義する場合にextendsを利用する。
(2) extendsの後ろには，スーパクラスの名前を一つだけ指定できる。
(3) サブクラスからインスタンスを生成すると，スーパクラスに定義されたインスタンス変数やメソッドがこのインスタンス内部に引き継がれる。（**継承**）

2．予約語 super

(1) サブクラスの“コンストラクタの処理内容”や“メソッドの処理内容”を記述する場合にsuperを利用できる。
(2)「super()」や「super.～」と記述して，サブクラスからスーパクラスに定義されたコンストラクタやメソッドを呼び出すことができる。

＜superの利用例＞

```
//図形クラス（スーパクラス）
class Figure {
   private String name;       //図形の名前
   public Figure(String n) {
      name = n;
   }
   public void display() {
      System.out.println("図形：" + name);
   }
}
//正方形クラス（サブクラス）
class Square extends Figure {
   private double length;     //一辺の長さ
   public Square(String n, double len) {
      super(n);
      length = len;
   }
   public void display() {
      super.display();
      System.out.println("一辺の長さ：" + length);
   }
}
```

スーパクラスの
コンストラクタを
呼び出す

スーパクラスの
displayメソッドを
呼び出す

3. 予約語 this

(1) クラスの“コンストラクタの処理内容”や“メソッドの処理内容”を記述する場合にthisを利用できる。

(2)「this()」や「this.～」と記述して，同じクラスに定義されたコンストラクタを呼び出したり，インスタンス変数を指定することができる。

＜thisの利用例＞

```
//図形クラス（スーパクラス）
class Figure {
   private String name;       //図形の名前
   public Figure() {
      this("名無し");
   }
   public Figure(String name) {
      this.name = name;
   }
   public void display() {
      System.out.println("図形：" + name);
   }
}
//正方形クラス（サブクラス）
class Square extends Figure {
   private double length;     //一辺の長さ
   public Square(String name, double length) {
      super(name);
      this.length = length;
   }
   public void display() {
      super.display();
      System.out.println("一辺の長さ：" + length);
   }
}
```

同じクラスのコンストラクタを呼び出す

インスタンス変数nameを指定する

インスタンス変数lengthを指定する

☆ミスしやすいJava文法

super()

サブクラスのコンストラクタにおいて，super()を処理内容の先頭行に記述しなければならない。また，super()によって呼び出されるコンストラクタがスーパクラスに定義されていなければならない。

this()

コンストラクタにおいて，this()を処理内容の先頭行に記述しなければならない。また，this()によって呼び出されるコンストラクタが同じクラスに定義されていなければならない。

問 1−1

次のJavaプログラムの説明及びプログラムを読んで，プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

〔プログラムの説明〕

あるアプリケーションのプロセス管理機能のプログラムである。

(1) 抽象クラスBasicProcessは，基本的な処理を表す。フィールドprocessは処理名を保持する。
　① コンストラクタは，引数で与えられた処理名をフィールドprocessに保持する。
　② メソッドgetProcessは，処理名を返す。
　③ 抽象メソッドexecuteは，処理名を表示する。

(2) クラスLoginProcessは，抽象クラスBasicProcessを拡張するクラスであり，ログイン処理を表す。
　① コンストラクタは，引数で与えられた処理名をフィールドprocessに保持する。
　② メソッドexecuteは，処理名を表示する。

(3) クラスLoginProcessTesterはテスト用のプログラムである。実行結果を図１に示す。

```
＜ログイン処理実行＞
```

図１　実行結果

〔プログラム１〕

```
public abstract class BasicProcess {
   private String process;

   public BasicProcess(String process) {
      this.process = process;
   }

   public String getProcess() {
      return process;
   }

   public abstract void execute();
}
```

〔プログラム２〕

```
public class LoginProcess [ a ] BasicProcess {
    public LoginProcess(String process) {
        [ b ];
    }

    public void execute() {
        System.out.println("＜" + getProcess() + "処理実行＞");
    }
}
```

〔プログラム３〕

```
public class LoginProcessTester {
    public static void main(String[] args) {
        LoginProcess process = new LoginProcess("ログイン");
        process.execute();
    }
}
```

a に関する解答群

ア　abstract	イ　extends
ウ　implements	エ　throws

b に関する解答群

ア　super()	イ　super(process)
ウ　this()	エ　this(process)

☆抽象クラス

・抽象クラスには，インスタンス変数，static変数，定数，コンストラクタ，（具象）メソッド，抽象メソッドを定義することができる。
・抽象クラスを定義する場合は，クラスに**abstract**を記述する。
・抽象メソッドを定義する場合，メソッドに**abstract**を記述する。

☆フィールド

・インスタンス変数，static変数，定数を**フィールド**や**メンバ変数**と呼ぶ。

問 1−2

次のJavaプログラムの説明及びプログラムを読んで，プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

〔プログラムの説明〕

ある企業の部署管理システムのプログラムである。

(1) クラスDepartmentは，部署情報を表す。フィールドdeptNoは部署番号，フィールドdeptNameは部署名を保持する。
　① 引数を取らないコンストラクタは，引数を取るコンストラクタを利用して部署番号0と部署名“未設定”をフィールドに保持する。
　② 引数を取るコンストラクタは，引数で与えられた部署番号と部署名をフィールドに保持する。
　③ メソッドdisplayは，部署情報を表示する。
(2) クラスAccountingは，経理部情報を表す。フィールドcostは経費を保持する。
　① コンストラクタは，引数で与えられた部署番号，部署名，経費をフィールドに保持する。
　② メソッドsetCostは，引数で与えられた経費をフィールドに保持する。
　③ メソッドdisplayは，経理部情報を表示する。
(3) クラスPersonnelは，人事部情報を表す。フィールドnumberは社員数を保持する。
　① コンストラクタは，引数で与えられた部署番号，部署名，社員数をフィールドに保持する。
　② メソッドsetNumberは，引数で与えられた社員数をフィールドに保持する。
　③ メソッドdisplayは，人事部情報を表示する。
(4) クラスDepartmentTesterはテスト用のプログラムである。実行結果を図1に示す。

```
＜部署情報＞
部署番号: 301
部署名　: 経理部
経費　　: 0
部署番号: 201
部署名　: 人事部
社員数　: 50
＜部署情報変更＞
部署番号: 301
部署名　: 経理部
経費　　: 1,000,000
部署番号: 201
部署名　: 人事部
社員数　: 70
```

図1　実行結果

〔プログラム１〕

```
public class Department {
    private int deptNo;
    private String deptName;

    public Department() {
        [ a ];
    }

    public Department(int deptNo, String deptName) {
        this.deptNo = deptNo;
        this.deptName = deptName;
    }

    public void display() {
        System.out.printf(" 部署番号: %s%n", deptNo);
        System.out.printf(" 部署名   : %s%n", deptName);
    }
}
```

〔プログラム２〕

```
public class Accounting extends Department {
    private int cost;

    public Accounting(int deptNo, String deptName, int cost) {
        super(deptNo, deptName);
        this.cost = cost;
    }

    public void setCost(int cost) {
        this.cost = cost;
    }

    public void display() {
        [ b ];
        System.out.printf(" 経費     : %,d%n", cost);
    }
}
```

〔プログラム３〕

```
public class Personnel extends Department {
   private int number;

   public Personnel(int deptNo, String deptName, int number) {
      super(deptNo, deptName);
      this.number = number;
   }

   public void setNumber(int number) {
      this.number = number;
   }

   public void display() {
      [          ];            //網掛け部分には適切なコードが記載されている
      System.out.printf(" 社員数  : %d%n", number);
   }
}
```

〔プログラム４〕

```
public class DepartmentTester {
   public static void main(String[] args) {
      System.out.println("＜部署情報＞");
      Accounting dept1 = new Accounting(301, "経理部", 0);
      Personnel dept2 = new Personnel(201, "人事部", 50);
      dept1.display();
      dept2.display();
      System.out.println("＜部署情報変更＞");
      dept1.[  c  ];
      dept2.[  d  ];
      dept1.display();
      dept2.display();
   }
}
```

a に関する解答群

ア　super()
イ　super(0, "未設定")
ウ　this()
エ　this(0, "未設定")

bに関する解答群
ア　display()
イ　super()
ウ　super(deptNo, deptName)
エ　super.display()

cに関する解答群
ア　Accounting(301, "経理部", 1000000)
イ　display()
ウ　setCost(1000000)
エ　setNumber(1000000)

dに関する解答群
ア　display()
イ　Personnel(201, "人事部", 70)
ウ　setCost(70)
エ　setNumber(70)

☆System.out.printfメソッド
・System.out.printfメソッドは，( )内に指定された書式指定にしたがって値を表示する。
＜利用例＞

```
String item = "基本情報STEP UP演習 Java対策";
int price = 1300;
System.out.printf("商品名：「%s」税込価格：%,d円%n", item, price);
```

値を埋め込んで表示する

書式指定には，次の要素とエスケープシーケンスを指定することができる。

＜基本要素＞
%d … 整数
%f … 実数
%c … 文字
%b … 真偽値
%s … 文字列

＜追加要素＞
, … ３桁区切り
. … 小数点
- … 左揃え
+ … ＋記号印字
0 … 上位桁０印字
% … %記号印字
n … 改行

＜エスケープシーケンス＞
¥n … 改行コード
¥t … タブ記号のコード
¥¥ … ¥記号のコード
¥’ … ’記号のコード
¥" … "記号のコード