＜実装＞

ルールを作り、それに沿ってクラスを作成していくことです。

クラス（class）　　→　処理をプログラミング
インターフェイス（interface）　→　ルールをプログラミング
＊eclipse の処理：クラスを作るときと同様に、「新規作成」から「インターフェイス」を選択して作成できます。

（演習①）
Java プロジェクト「Iphone2」を作成しましょう。
以下の Mp3Player インターフェイスを作成後、プログラミングしましょう。

```
public interface Mp3Player {

  public abstract void play();

  public abstract void stop();

  public abstract void next();

  public abstract void back();

}
```

interface：イメージとしてはこれからルールを作りますよという宣言します。

abstract を使ってそれぞれのルール（メソッド名のみ）を作っています。

interface と abstract はセットで使います。
メソッド名のみで処理は書かない不思議な形で。

メソッド名のみで処理内容は書かないメソッドの事を**「抽象メソッド」**といいます。
具体的な処理を書かないので「抽象的な」メソッドです。
このため「abstract（抽象的な）」というキーワードをつけます。

（演習②）

MP3Player というインターフェイスを実装して SmartPhone クラスを作ります。

以下の SmartPhone クラスを作成後、プログラミングしましょう。

```
public class SmartPhone implements Mp3Player {
  public void play(){
    System.out.println("再生");
  }
  public void stop(){
    System.out.println("停止");
  }
  public void next(){
    System.out.println("次へ");
  }
  public void back(){
    System.out.println("戻る");
  }
}
```

クラスにインターフェイスを実装するには implements を指定しなくてはいけません。また Mp3Player で作ったルール（メソッド）は必ず SmartPhone クラスで上書きし、処理内容を記述する必要があります。

※必ずインターフェースに書いたメソッドの処理内容を書く

Mp3Player がインタフェースなので、必ず SmartPhone クラスには Mp3Player で作ったルール（メソッド）と同じメソッドを書き、さらに処理内容を書かなくてはいけません。

（処理の内容に関しては自由です。ここでいうと「System.out.println("再生");」の部分は何を書いても大丈夫です。

（演習③）

以下の Iphone クラスを作成後、プログラミングして実行してみましょう。

```
public class Iphone{
  public static void main(String[] args){
```

```
    SmartPhone iphone = new SmartPhone();
    iphone.play();
    iphone.stop();
    iphone.next();
    iphone.back();
  }
}
```

＜実装の応用＞

①implements の後ろにインターフェイスは複数追加することが可能です。

例)

```
public class XXXXX implements AAAAA, BBBBB, CCCCC ・・・・・ {

}
```

インターフェイス

②継承と組み合わせることができます。

（演習④）

NewFunction インターフェイスを作成してみましょう。

```
public interface NewFunction {

  public abstract void call();

  public abstract void mail();

  public abstract void photo();

  public abstract void internet();

}
```

抽象メソッド

（演習⑤）

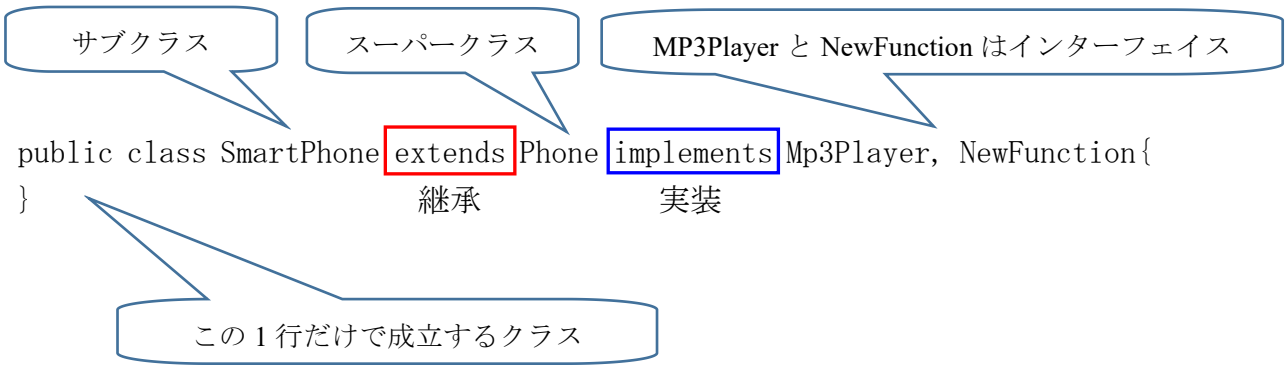
Phone クラスを作成してみましょう

```
public class Phone  {

  public void play(){
    System.out.println("再生");
  }
  public void stop(){
    System.out.println("停止");
  }  public void next(){
    System.out.println("次へ");
  }
  public void back(){
    System.out.println("戻る");
  }
  public void call(){
    System.out.println("電話");
  }
  public void mail(){
    System.out.println("メール");
  }
  public void photo(){
    System.out.println("写真");
  }
  public void internet(){
    System.out.println("インターネット");
  }

}
```

演習⑥ SmartPhone クラスを変更しましょう。



演習⑦ Iphone クラスをプログラミングをして実行してみましょう。

```
public class Iphone{
  public static void main(String[] args){
    SmartPhone iphone = new SmartPhone();
    iphone.play();
    iphone.stop();
    iphone.next();
    iphone.back();
    iphone.call();
    iphone.mail();
    iphone.photo();
    iphone.internet();
  }
}
```