

Java



상속과 인터페이스

클래스의 상속에 대하여
인터페이스에 대하여

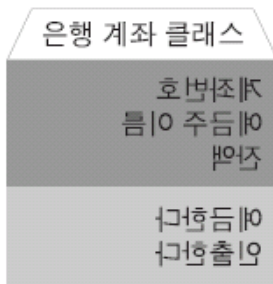


상속과 인터페이스

01. 클래스의 상속

언제 상속이 필요한가?

- 기존 클래스와 유사한 클래스를 만들어야 할 경우



계좌번호: 555-666-77777777
예금주 이름: 박진희
잔액: 1,124,021원

예금한다
인출한다

은행 계좌 클래스로 만든 객체

계좌번호: 555-666-77777777
예금주 이름: 박진희
잔액: 1,124,021원
직불카드 번호: 1111222233334444

예금한다
인출한다
직불카드 사용액을 지불한다

은행 계좌 객체와
유사한 형태의 객체가 필요



상속과 인터페이스

01. 클래스의 상속

상속이란?

• 상속(inheritance) : 기존 클래스를 확장해서 새로운 클래스를 만드는 기술

[이름] 은행 계좌 클래스
[데이터] 계좌번호 예금주 이름 잔액
[기능] 예금한다 인출한다

[데이터] 직불카드 번호
[기능] 직불카드 사용액을 지불한다

[이름] 직불 계좌 클래스
[데이터] 계좌번호 예금주 이름 잔액 직불카드 번호
[기능] 예금한다 인출한다 직불카드 사용액을 지불한다



기존에 있던 클래스

+



새로 만들어진 클래스

실제로
작성해야 하는 부분

상속과 인터페이스

01. 클래스의 상속

클래스 상속의 기초 문법

• [예제 6-1] 은행 계좌 클래스

```
1  class Account {  
2      String accountNo;  
3      String ownerName;  
4      int balance;  
5      void deposit(int amount) {  
6          balance += amount;  
7      }  
8      int withdraw(int amount) throws Exception {  
9          if (balance < amount)  
10             throw new Exception("잔액이 부족합니다.");  
11             balance -= amount;  
12             return amount;  
13     }  
14 }
```

[이름] 은행 계좌 클래스

[데이터]
계좌번호
예금주 이름
잔액

[기능]
예금한다
인출한다

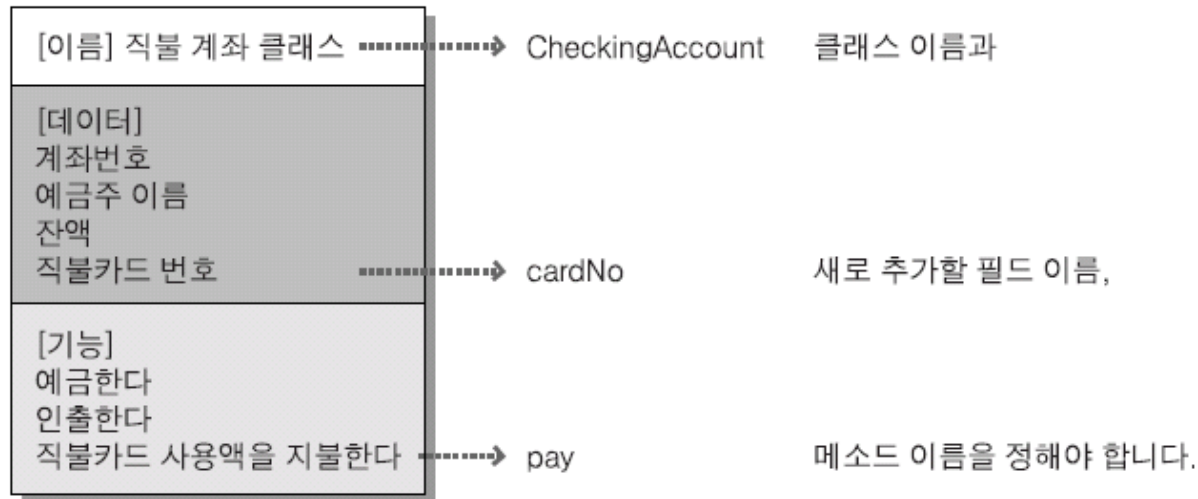
상속과 인터페이스

01. 클래스의 상속

클래스 상속의 기초 문법

• 다른 클래스를 상속하는 클래스의 선언

- - 제일 먼저 해야 할 일은 정의된 클래스와 추가되는 필드, 메소드 이름을 정하는 것



상속과 인터페이스

01. 클래스의 상속

클래스 상속의 기초 문법

- 다른 클래스를 상속하는 클래스의 선언
 - extends 키워드를 이용하여 상속할 클래스 이름을 명시해야 함

```
class CheckingAccount extends Account {  
    ...  
}
```

자바 키워드

상속할 클래스 이름

상속과 인터페이스

01. 클래스의 상속

클래스 상속의 기초 문법

- [예제 6-2] 은행 계좌 클래스를 상속하는 직불 계좌 클래스

```
1  class CheckingAccount extends Account {
2      String cardNo;
3      int pay(String cardNo, int amount) throws Exception {
4          if (!cardNo.equals(this.cardNo) || (balance < amount)) {
5              throw new Exception("지불이 불가능합니다.");
6          }
7      }
8  }
```

extends 절

직불카드 번호에 해당하는 필드

직불카드 사용액을 지불한다에 해당하는 메소드

```
명령 프롬프트

E:\work\chap6\6-1-1>javac Account.java

E:\work\chap6\6-1-1>javac CheckingAccount.java

E:\work\chap6\6-1-1>
```


상속과 인터페이스

클래스 상속의 기초 문법

- [예제 6-3] 직불 계좌 클래스를 사용하는 프로그램

```
1  class InheritanceExample1 {
2      public static void main(String args[]) {
3          CheckingAccount obj = new CheckingAccount();
4          obj.accountNo = "111-22-33333333";
5          obj.ownerName = "홍길동";
6          obj.cardNo = "5555-6666-7777-8888";
7          obj.deposit(100000);
8          try {
9              int paidAmount = obj.pay("5555-6666-7777-8888", 47000);
10             System.out.println("지불액:" + paidAmount);
11             System.out.println("잔액:" + obj.balance);
12         }
13         catch (Exception e) {
14             String msg = e.getMessage();
15             System.out.println(msg);
16         }
17     }
18 }
```

Account 클래스로부터
상속받은 필드 사용

Account 클래스로부터
상속받은 메소드 호출

```
명령 프롬프트

E:\work\chap6\6-1-1>javac InheritanceExample1.java

E:\work\chap6\6-1-1>java InheritanceExample1
지불액:47000
잔액:53000

E:\work\chap6\6-1-1>
```

상속과 인터페이스

상속과 생성자

- [예제 6-4] 생성자가 추가된 직불 계좌 클래스

```
1  class CheckingAccount extends Account {
2      String cardNo;
3      CheckingAccount(String accountNo, String ownerName,
4                      int balance, String cardNo) { // 생성자
5          this.accountNo = accountNo;
6          this.ownerName = ownerName;
7          this.balance = balance;
8          this.cardNo = cardNo; ----- 클래스 안에 선언된 필드를 초기화합니다.
9      }
10     int pay(String cardNo, int amount) throws Exception {
11         if (!cardNo.equals(this.cardNo) || (balance < amount))
12             throw new Exception("지불이 불가능합니다.");
13         return withdraw(amount);
14     }
```

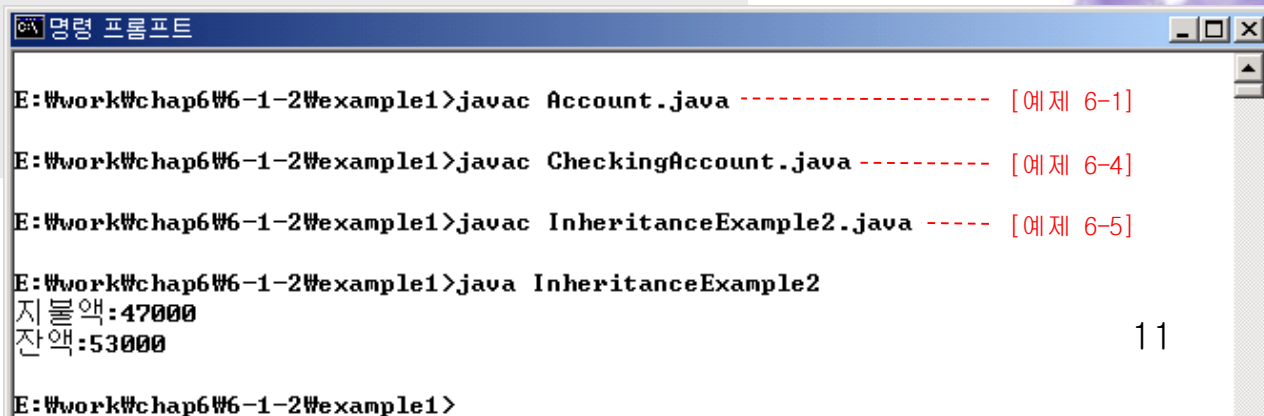
슈퍼클래스로부터 상속받은 필드들을 초기화합니다.

상속과 인터페이스

상속과 생성자

- [예제 6-5] 직불 계좌 클래스의 생성자를 사용하는 프로그램

```
1  class InheritanceExample2 {
2      public static void main(String args[]) {
3          CheckingAccount obj = new CheckingAccount("111-22-33333333",
4              "홍길동", 0, "5555-6666-7777-8888"); ----- CheckingAccount
5          obj.deposit(100000);                               클래스의 생성자 호출
6          try {
7              int paidAmount = obj.pay("5555-6666-7777-8888", 47000);
8              System.out.println("지불액:" + paidAmount);
9              System.out.println("잔액:" + obj.balance);
10         }
11         catch (Exception e) {
12             String msg = e.getMessage();
13             System.out.println(msg);
14         }
15     }
```



```
명령 프롬프트

E:\work\chap6\6-1-2\example1>javac Account.java ----- [예제 6-1]

E:\work\chap6\6-1-2\example1>javac CheckingAccount.java ----- [예제 6-4]

E:\work\chap6\6-1-2\example1>javac InheritanceExample2.java ----- [예제 6-5]

E:\work\chap6\6-1-2\example1>java InheritanceExample2
지불액:47000
잔액:53000

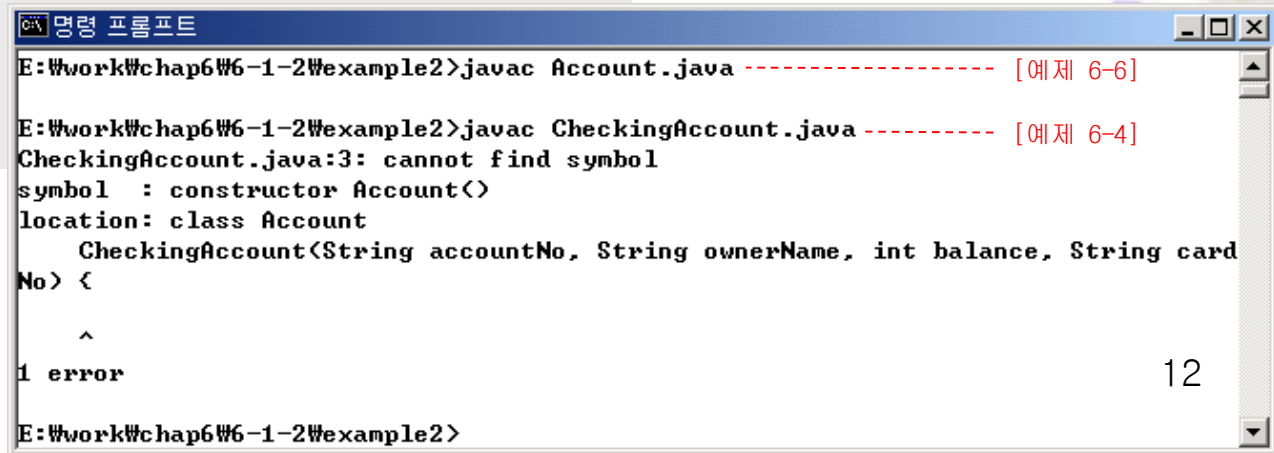
E:\work\chap6\6-1-2\example1>
```

상속과 인터페이스

● 생성자가 있는 클래스의 상속

- [예제 6-6] 생성자가 있는 Account 클래스

```
1  class Account {
2      String accountNo;
3      String ownerName;
4      int balance;
5      Account(String accountNo, String ownerName, int balance) {
6          this.accountNo = accountNo;
7          this.ownerName = ownerName;
8          this.balance = balance;
9      }
10     void deposit(int amount) {
11         balance += amount;
12     }
13     int withdraw(int amount) throws Exception {
14         if (balance < amount)
15             throw new Exception("잔액이 부족합니다.");
16         balance -= amount;
17         return amount;
18     }
19 }
```



```
명령 프롬프트
E:\work\chap6\6-1-2\example2>javac Account.java ----- [예제 6-6]
E:\work\chap6\6-1-2\example2>javac CheckingAccount.java ----- [예제 6-4]
CheckingAccount.java:3: cannot find symbol
symbol  : constructor Account()
location: class Account
    CheckingAccount(String accountNo, String ownerName, int balance, String card
No> {
    ^
1 error
E:\work\chap6\6-1-2\example2>
```

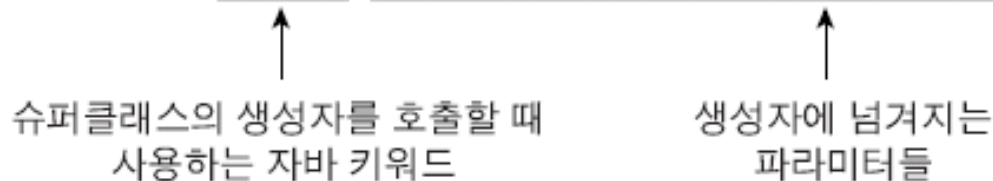
01. 클래스의 상속

생성자가 있는 클래스의 상속

- [예제 6-6], [예제 6-4]를 컴파일할 때 에러가 발생한 이유
 - 자바 컴파일러는 생성자 안의 첫번째 명령문이 슈퍼클래스의 생성자 호출문이 아니면 자동으로 슈퍼클래스의 파라미터 없는 생성자(no-arg constructor) 호출문을 추가하기 때문

.. 슈퍼클래스의 생성자 호출문 작성 방법

```
super(accountNo, ownerName, balance);
```

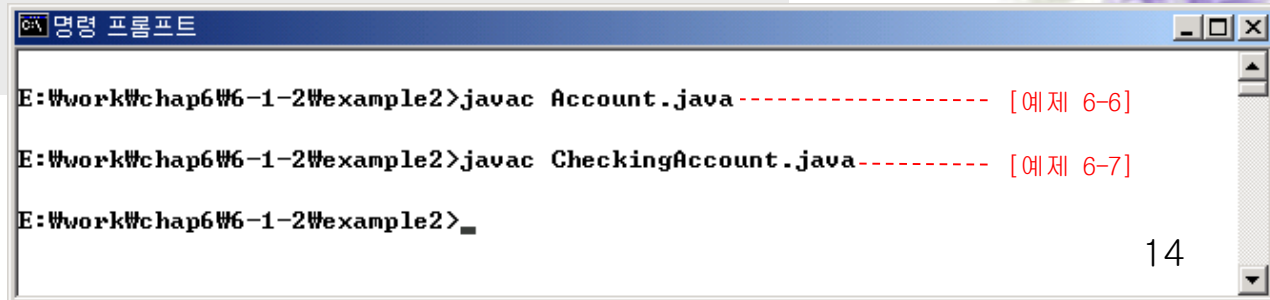


상속과 인터페이스

● 생성자가 있는 클래스의 상속

- [예제 6-7] 슈퍼클래스의 생성자를 호출하는 직불 계좌 클래스

```
1  class CheckingAccount extends Account {
2      String cardNo;
3      CheckingAccount(String accountNo, String ownerName,
4                      int balance, String cardNo) {
5          super(accountNo, ownerName, balance);----- 슈퍼클래스의 생성자 호출
6          this.cardNo = cardNo;
7      }
8      int pay(String cardNo, int amount) throws Exception {
9          if (!cardNo.equals(this.cardNo) || (balance < amount))
10             throw new Exception("지불이 불가능합니다.");
11             return withdraw(amount);
12     }
```



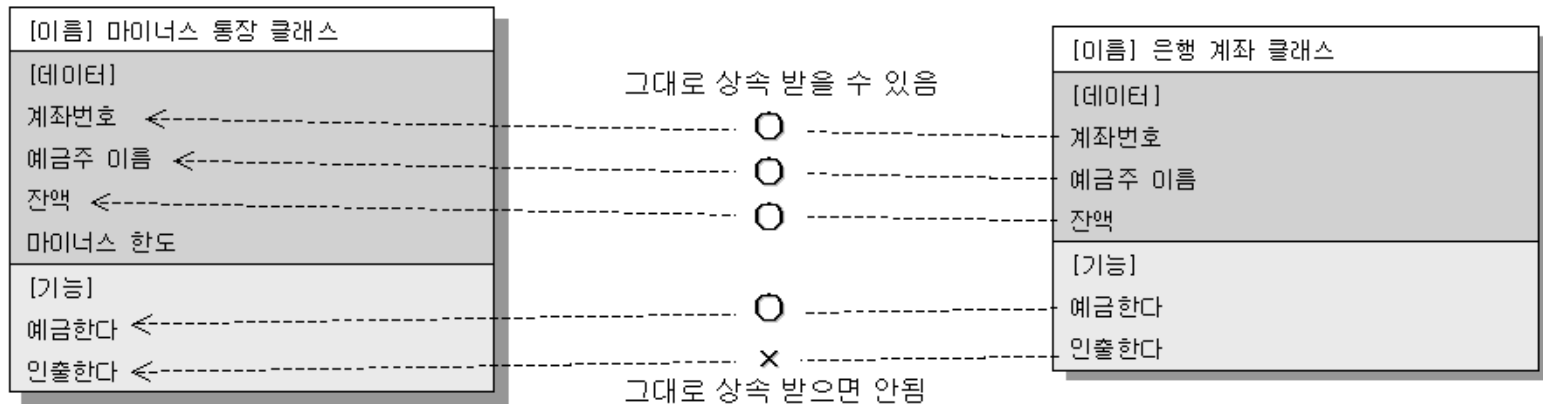
```
명령 프롬프트
E:\work\chap6\6-1-2\example2>javac Account.java----- [예제 6-6]
E:\work\chap6\6-1-2\example2>javac CheckingAccount.java----- [예제 6-7]
E:\work\chap6\6-1-2\example2>_
```

상속과 인터페이스

01. 클래스의 상속

메소드 오버라이딩

- 메소드 오버라이딩이 필요한 경우



- 메소드 오버라이딩(method overriding)의 방법
 - 슈퍼클래스와 똑같은 시그니처를 갖는 메소드를 서브클래스에 선언

상속과 인터페이스

메소드 오버라이딩

- [예제 6-8] 슈퍼클래스의 메소드를 오버라이드하는 마이너스 통장 클래스

```
1  class CreditLineAccount extends Account {
2      int creditLine; ----- 마이너스 한도 필드
3      CreditLineAccount(String accountNo, String ownerName,
4                          int balance, int creditLine) {
5          super(accountNo, ownerName, balance);
6          this.creditLine = creditLine;
7      }
8      int withdraw(int amount) throws Exception {
9          if ((balance + creditLine) < amount)
10             throw new Exception("인출이 불가능합니다.");
11             balance -= amount;
12             return amount;
13     }
```

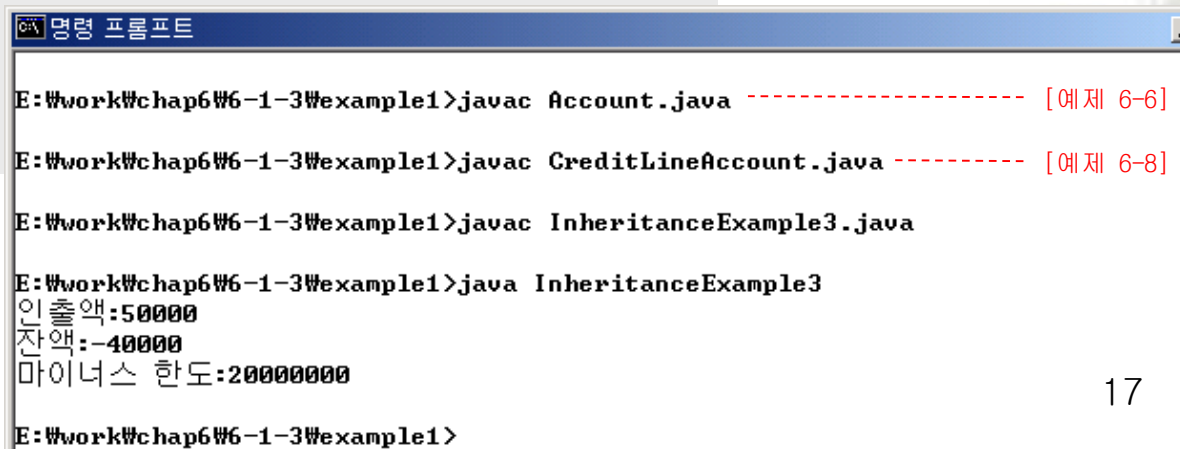
인출한다 기능을
다시 구현하는 메소드

상속과 인터페이스

메소드 오버라이딩

- [예제 6-9] 마이너스 통장 클래스를 사용하는 프로그램

```
1  class InheritanceExample3 {
2      public static void main(String args[]) {
3          CreditLineAccount obj = new CreditLineAccount(
4              "000-11-111111", "김선달", 10000, 20000000);
5          try {
6              int amount = obj.withdraw(50000);
7              System.out.println("인출액:" + amount);
8              System.out.println("잔액:" + obj.balance);
9              System.out.println("마이너스 한도:" + obj.creditLine);
10         }
11         catch (Exception e) {
12             System.out.println(e.getMessage());
13         }
14     }
```



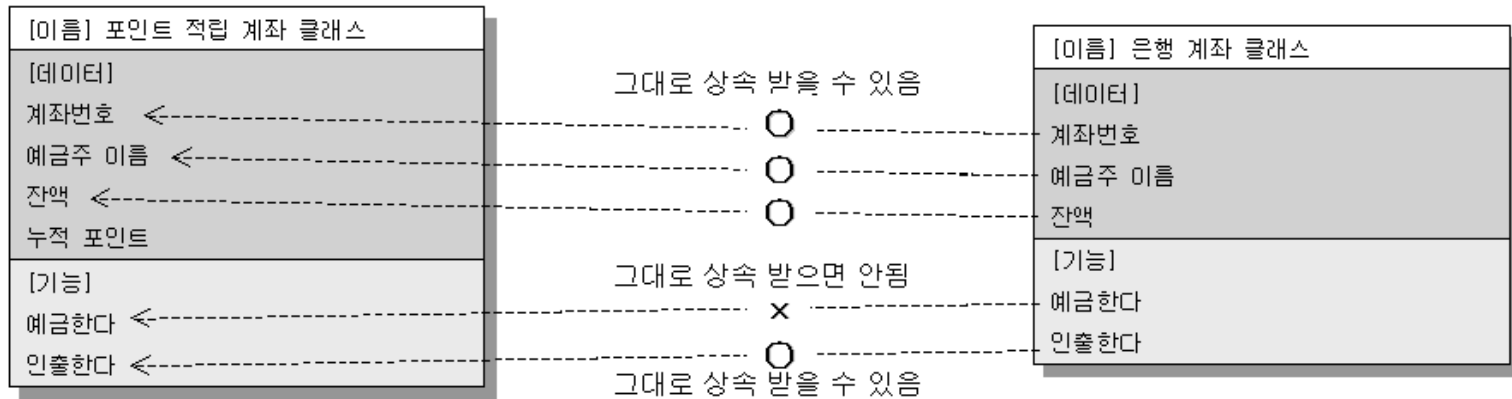
```
E:\work\chap6\6-1-3\example1>javac Account.java ----- [예제 6-6]
E:\work\chap6\6-1-3\example1>javac CreditLineAccount.java ----- [예제 6-8]
E:\work\chap6\6-1-3\example1>javac InheritanceExample3.java
E:\work\chap6\6-1-3\example1>java InheritanceExample3
인출액:50000
잔액:-40000
마이너스 한도:20000000
E:\work\chap6\6-1-3\example1>
```

상속과 인터페이스

01. 클래스의 상속

메소드 오버라이딩

•• 메소드 오버라이딩이 필요한 또다른 경우의 예



상속과 인터페이스

01. 클래스의 상속

메소드 오버라이딩

- [예제 6-10] 슈퍼클래스의 메소드를 오버라이드하는 포인트 적립 계좌 클래스

```
1    class BonusPointAccount extends Account {  
2        int bonusPoint;    // 누적 포인트  
3        BonusPointAccount(String accountNo, String ownerName, int balance,  
4                               int bonusPoint) {  
5            super(accountNo, ownerName, balance);  
6            this.bonusPoint = bonusPoint;  
7        }  
8        void deposit(int amount) {    // 예금한다  
9            balance += amount;  
10           bonusPoint += (int) (amount * 0.001);  
11        }  
12    }
```

상속과 인터페이스

01. 클래스의 상속

메소드 오버라이딩

- 슈퍼클래스에 있는 오버라이드된 메소드를 호출하는 방법

`super.deposit(amount);`

호출하는 메소드가
슈퍼클래스의 메서드임을
가리키는 자바 키워드

메소드 이름

파라미터

상속과 인터페이스

01. 클래스의 상속

메소드 오버라이딩

- [예제 6-11] 오버라이드된 메소드를 호출하는 포인트 적립 계좌 클래스

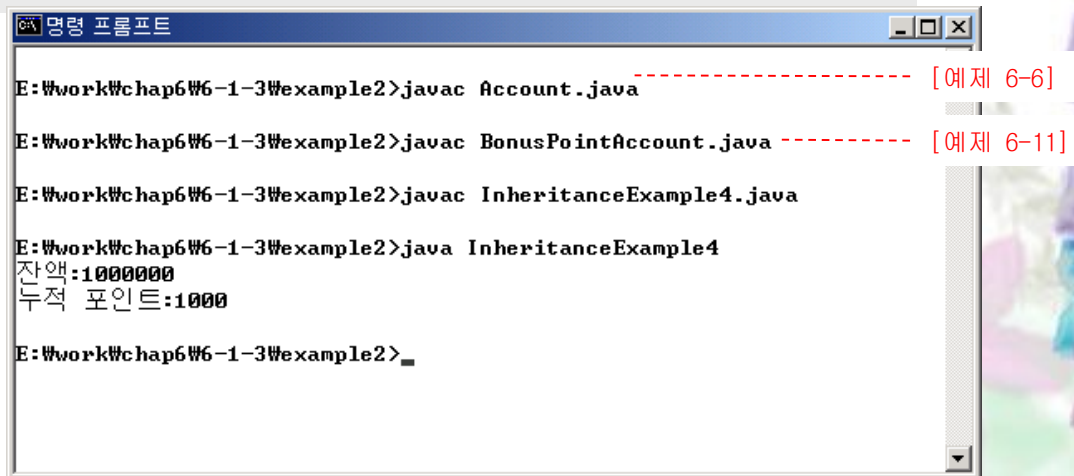
```
1    class BonusPointAccount extends Account {
2        int bonusPoint;    // 누적 포인트
3        BonusPointAccount(String accountNo, String ownerName, int balance,
4                               int bonusPoint) {
5            super(accountNo, ownerName, balance);
6            this.bonusPoint = bonusPoint;
7        }
8        void deposit(int amount) {    // 예금한다
9            super.deposit(amount); ----- 슈퍼클래스의 deposit 메소드 호출
10           bonusPoint += (int) (amount * 0.001);
11        }
```

상속과 인터페이스

메소드 오버라이딩

- [예제 6-12] 포인트 적립 계좌 클래스를 사용하는 프로그램

```
1  class InheritanceExample4 {  
2      public static void main(String args[]) {  
3          BonusPointAccount obj = new BonusPointAccount(  
4              "333-33-333333", "김미영", 0, 0);  
5          obj.deposit(1000000);  
6          System.out.println("잔액:" + obj.balance);  
7          System.out.println("누적 포인트:" + obj.bonusPoint);  
8      }  
9  }
```



```
명령 프롬프트  
E:\work\chap6\6-1-3\example2>javac Account.java  
E:\work\chap6\6-1-3\example2>javac BonusPointAccount.java  
E:\work\chap6\6-1-3\example2>javac InheritanceExample4.java  
E:\work\chap6\6-1-3\example2>java InheritanceExample4  
잔액:1000000  
누적 포인트:1000  
E:\work\chap6\6-1-3\example2>
```

상속과 인터페이스

상속을 금지하는 final 키워드

- [예제 6-13] final 키워드를 추가한 Account 클래스

```
----- final 키워드 추가
1  final class Account {
2      String accountNo;
3      String ownerName;
4      int balance;
5      Account(String accountNo, String ownerName, int balance) {
6          this.accountNo = accountNo;
7          this.ownerName = ownerName;
8          this.balance = balance;
9      }
10     void deposit(int amount) {
11         balance += amount;
12     }
13     int withdraw(int amount) throws Exception {
14         if (balance < amount)
15             throw new Exception("잔액이 부족합니다.");
16         balance -= amount;
17         return amount;
18     }
19 }
```

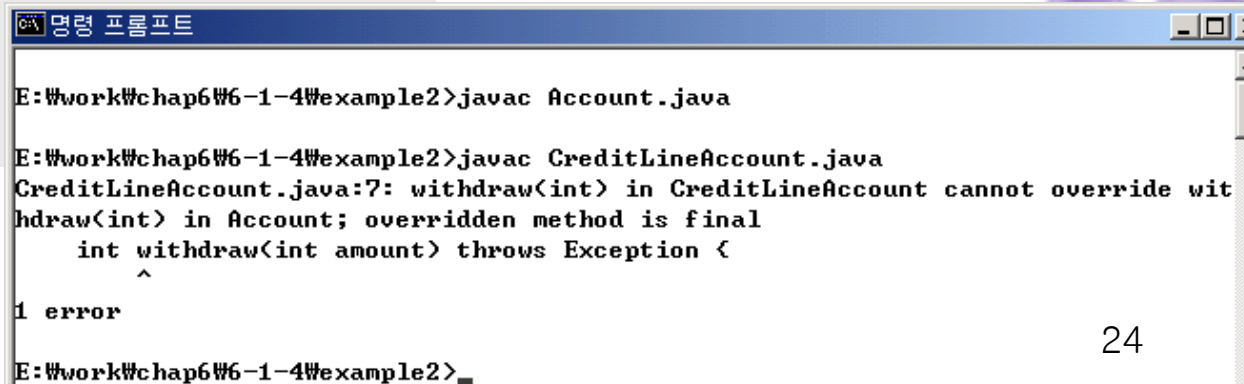
```
명령 프롬프트
E:\work\chap6\6-1-4\example1>javac Account.java
E:\work\chap6\6-1-4\example1>javac BonusPointAccount.java ----- [예제 6-11]
BonusPointAccount.java:1: cannot inherit from final Account
class BonusPointAccount extends Account {
                               ^
1 error
E:\work\chap6\6-1-4\example1>_
```

상속과 인터페이스

메소드 오버라이딩을 금지하는 final 키워드

- [예제 6-14] withdraw 메소드에 final 키워드를 추가한 Account 클래스

```
1 class Account {
2     String accountNo;
3     String ownerName;
4     int balance;
5     Account(String accountNo, String ownerName, int balance) {
6         this.accountNo = accountNo;
7         this.ownerName = ownerName;
8         this.balance = balance;
9     }
10    void deposit(int amount) {
11        balance += amount;
12    } ----- final 키워드 추가
13    final int withdraw(int amount) throws Exception {
14        if (balance < amount)
15            throw new Exception("잔액이 부족합니다.");
16        balance -= amount;
17        return amount;
18    }
19 }
```



명령 프롬프트 창에서 실행된 명령어와 출력 결과:

```
E:\work\chap6\6-1-4\example2>javac Account.java
E:\work\chap6\6-1-4\example2>javac CreditLineAccount.java
CreditLineAccount.java:7: withdraw(int) in CreditLineAccount cannot override wit
hdraw(int) in Account; overridden method is final
    int withdraw(int amount) throws Exception {
        ^
1 error
E:\work\chap6\6-1-4\example2>_
```


상속과 인터페이스

01. 클래스의 상속

인스턴스화를 금지하는 abstract 키워드

- [예제 6-15] abstract 키워드를 추가한 Account 클래스

```
1  abstract class Account {  
2      String accountNo;  
3      String ownerName;  
4      int balance;  
5      Account(String accountNo, String ownerName, int balance) {  
6          this.accountNo = accountNo;  
7          this.ownerName = ownerName;  
8          this.balance = balance;  
9      }  
10     void deposit(int amount) {  
11         balance += amount;  
12     }  
13     int withdraw(int amount) throws Exception {  
14         if (balance < amount)  
15             throw new Exception("잔액이 부족합니다.");  
16         balance -= amount;  
17         return amount;  
18     }  
19 }
```

abstract 키워드 추가

abstract 키워드가 붙은 클래스를
추상 클래스(abstract class)라고 함

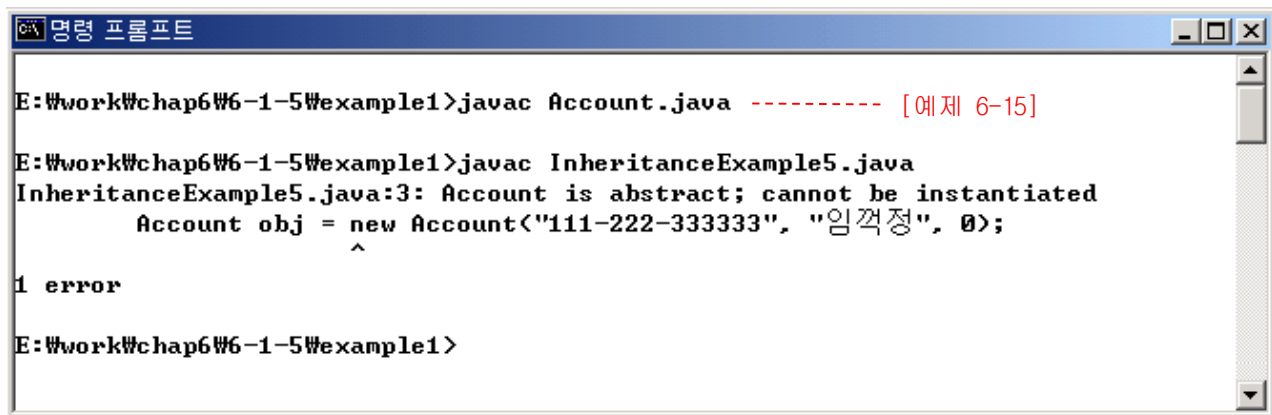
상속과 인터페이스

01. 클래스의 상속

인스턴스화를 금지하는 abstract 키워드

- [예제 6-16] 추상 클래스를 인스턴스화하는 잘못된 프로그램

```
1  class InheritanceExample5 {  
2      public static void main(String args[]) {  
3          Account obj = new Account("111-222-333333", "임꺽정", 0);  
4      }  
5  }
```



```
명령 프롬프트  
E:\work\chap6\6-1-5\example1>javac Account.java ----- [예제 6-15]  
E:\work\chap6\6-1-5\example1>javac InheritanceExample5.java  
InheritanceExample5.java:3: Account is abstract; cannot be instantiated  
    Account obj = new Account("111-222-333333", "임꺽정", 0);  
                        ^  
1 error  
E:\work\chap6\6-1-5\example1>
```

상속과 인터페이스

01. 클래스의 상속

추상 메소드

- 추상 메소드(*abstract method*) : 메소드 본체가 없는 메소드
- 추상 메소드가 필요한 경우

[이름] 메시지 발송 클래스
[데이터] 제목 발송자 이름
[기능] 메시지를 송신한다



[이름] 이메일 송신 클래스
[데이터] 제목 발송자 이름 발송자 이메일 주소 이메일 내용
[기능] 메시지를 송신한다

[이름] 문자메시지 송신 클래스
[데이터] 제목 발송자 이름 회신 전화번호 메시지 본문
[기능] 메시지를 송신한다

상속과 인터페이스

01. 클래스의 상속

추상 메소드

- 추상 메소드의 선언 방법

```
abstract void sendMessage (String recipient);
```

본체가 없는 메소드를 선언할 때
반드시 써야 하는 자바 키워드

메소드의 리턴 타입, 이름,
파라미터 변수 선언

메소드 본체 대신
세미콜론을 써야 함

상속과 인터페이스

01. 클래스의 상속

추상 메소드

- [예제 6-17] 추상 클래스를 포함하는 클래스 – 메시지 발송 클래스

```
1  abstract class MessageSender {  
2      String title;  
3      String senderName;  
4      MessageSender(String title, String senderName) {  
5          this.title = title;  
6          this.senderName = senderName;  
7      }  
8      abstract void sendMessage(String recipient);  
9  }
```

클래스 자체도 추상 클래스로 선언

추상 메소드 선언

상속과 인터페이스

01. 클래스의 상속

추상 메소드

- [예제 6-18] 메시지 발송 클래스를 상속하는 이메일 송신 클래스

```
1  class EmailSender extends MessageSender {
2      String senderAddr;
3      String emailBody;
4      EmailSender(String title, String senderName,
5                  String senderAddr, String emailBody) {
6          super(title, senderName);
7          this.senderAddr = senderAddr;
8          this.emailBody = emailBody;
9      }
10     void sendMessage(String recipient) {
11         System.out.println("-----");
12         System.out.println("제목: " + title);
13         System.out.println("보내는 사람: " + senderName +
14                             " " + senderAddr);
15         System.out.println("받는 사람: " + recipient);
16         System.out.println("내용: " + emailBody);
17     }
18 }
```

} 슈퍼클래스의 메소드를
오버라이드하는 메소드

상속과 인터페이스

01. 클래스의 상속

추상 메소드

- [예제 6-19] 메시지 발송 클래스를 상속하는 문자 메시지 송신 클래스

```
1  class SMSSender extends MessageSender {
2      String returnPhoneNo;
3      String message;
4      SMSSender(String title, String senderName,
5                  String returnPhoneNo, String message) {
6          super(title, senderName);
7          this.returnPhoneNo = returnPhoneNo;
8          this.message = message;
9      }
10     void sendMessage(String recipient) {
11         System.out.println("-----");
12         System.out.println("제목: " + title);
13         System.out.println("보내는 사람: " + senderName);
14         System.out.println("전화번호: " + recipient);
15         System.out.println("회신 전화번호: " + returnPhoneNo);
16         System.out.println("메시지 내용: " + message);
17     }
18 }
```

} 슈퍼클래스의 메소드를
오버라이드하는 메소드

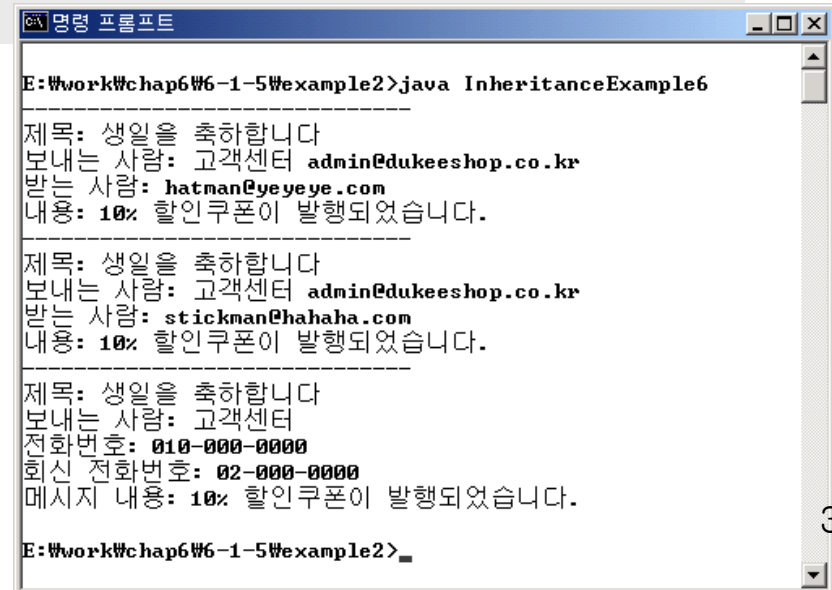
상속과 인터페이스

추상 메소드

- [예제 6-20] 이메일 송신 클래스와 문자 메시지 송신 클래스를 사용하는 프로그램

```
1 class InheritanceExample6 {  
2     public static void main(String args[]) {  
3         EmailSender obj1 = new EmailSender("생일을 축하합니다", "고객센터",  
4             "admin@dukeeshop.co.kr", "10% 할인쿠폰이 발행되었습니다.");  
5         SMSSender obj2 = new SMSSender("생일을 축하합니다", "고객센터",  
6             "02-000-0000", "10% 할인쿠폰이 발행되었습니다.");  
7         obj1.sendMessage("hatman@yeyeye.com");  
8         obj1.sendMessage("stickman@hahaha.com");  
9         obj2.sendMessage("010-000-0000");  
10    }  
11 }
```

추상 메소드를 구현하는 메소드를 호출합니다.



```
명령 프롬프트  
E:\work\chap6\6-1-5\example2>java InheritanceExample6  
-----  
제목: 생일을 축하합니다  
보내는 사람: 고객센터 admin@dukeeshop.co.kr  
받는 사람: hatman@yeyeye.com  
내용: 10% 할인쿠폰이 발행되었습니다.  
-----  
제목: 생일을 축하합니다  
보내는 사람: 고객센터 admin@dukeeshop.co.kr  
받는 사람: stickman@hahaha.com  
내용: 10% 할인쿠폰이 발행되었습니다.  
-----  
제목: 생일을 축하합니다  
보내는 사람: 고객센터  
전화번호: 010-000-0000  
회신 전화번호: 02-000-0000  
메시지 내용: 10% 할인쿠폰이 발행되었습니다.  
E:\work\chap6\6-1-5\example2>
```


상속과 인터페이스

01. 클래스의 상속

추상 메소드

- [예제 6-21] 메시지 발송 클래스를 상속하는 클래스 – 잘못된 예

```
1 class SillySender extends MessageSender {  
2     SillySender(String title, String senderName) {        // 생성자  
3         super(title, senderName);  
4     }  
5 }
```



```
C:\> 명령 프롬프트

E:\work\chap6\6-1-5\example2>javac MessageSender.java

E:\work\chap6\6-1-5\example2>javac SillySender.java
SillySender.java:1: SillySender is not abstract and does not override abstract method sendMessage<java.lang.String> in MessageSender
class SillySender extends MessageSender {
^
1 error

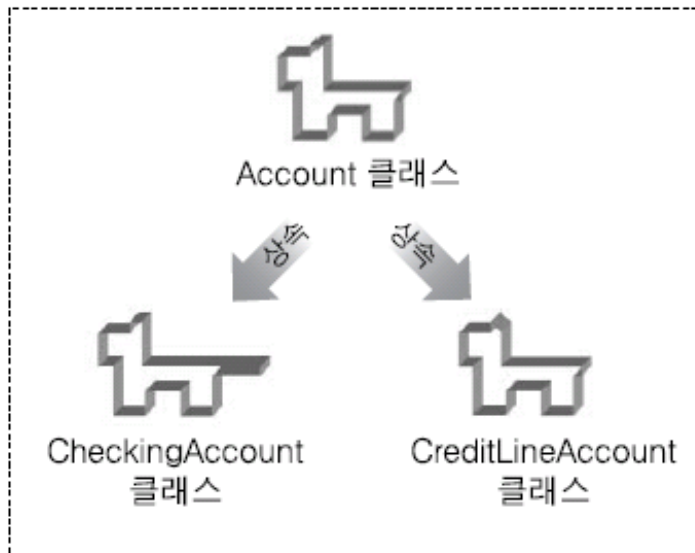
E:\work\chap6\6-1-5\example2>
```

상속과 인터페이스

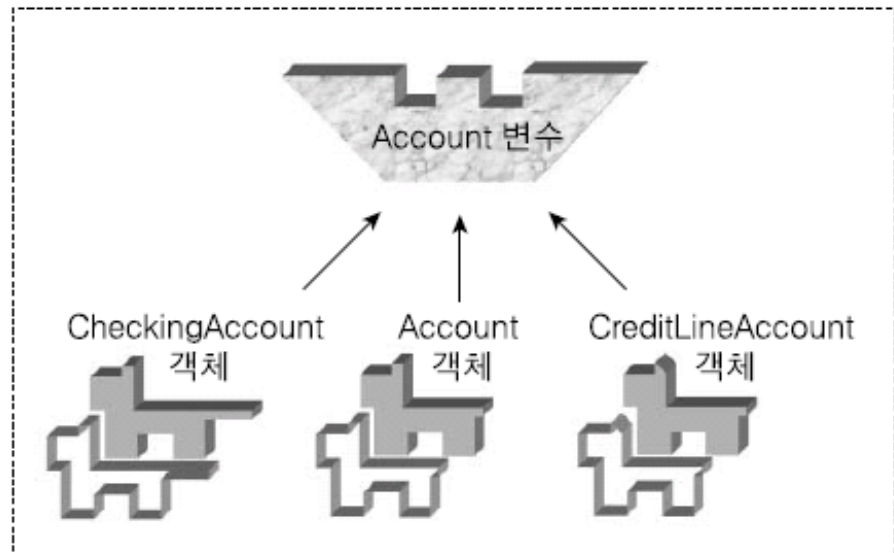
01. 클래스의 상속

클래스 변수의 다형성

• 클래스 변수의 다형성



상속을 통해 만들어진 클래스는 슈퍼클래스와 비슷한 양상을 띵니다.



그렇기 때문에 슈퍼클래스 변수에는 서브클래스 객체를 대입할 수 있습니다.

상속과 인터페이스

01. 클래스의 상속

클래스 변수의 다형성

• 클래스 변수의 다형성의 예

```
Account obj;  
obj = new Account("111-22-333333", "임꺽정", 10000);  
obj = new CheckingAccount("444-55-666666", "홍길동", 20000,  
                           "5555-6666-7777-8888");  
obj = new CreditLineAccount("777-88-999999", "김선달", 30000, 20000000);  
obj = new BonusPointAccount("000-00-000000", "김미영", 0, 0);
```

상속과 인터페이스

클래스 변수의 다형성

• [예제 6-22] 클래스 변수의 다형성을 활용하는 프로그램 (1)

```
1 class InheritanceExample7 {
2     public static void main(String args[]) {
3         Account obj1 = new Account("111-22-333333", "임꺽정", 10000);
4         CheckingAccount obj2 = new CheckingAccount("444-55-666666", "홍길동", 20000, "5555-6666-7777-8888");
5         CreditLineAccount obj3 = new CreditLineAccount("777-88-999999", "김선달", 30000, 20000000);
6         BonusPointAccount obj4 = new BonusPointAccount("000-00-000000", "김미영", 0, 0);
7         printAccountInfo(obj1);
8         printAccountInfo(obj2);
9         printAccountInfo(obj3);
10        printAccountInfo(obj4);
11    }
12    static void printAccountInfo(Account obj) {
13        System.out.println("계좌번호:" + obj.accountNo);
14        System.out.println("예금주 이름:" + obj.ownerName);
15        System.out.println("잔액:" + obj.balance);
16        System.out.println();
17    }
18 }
```

다양한 타입의 객체를 가지고 메소드를 호출합니다.

----- 다양한 타입의 객체를 한 타입의 파라미터 변수로 받습니다.

명령 프롬프트

E:\work\chap6\6-1-6\example1>java InheritanceExample7

계좌번호:111-22-333333
예금주 이름:임꺽정
잔액:10000

계좌번호:444-55-666666
예금주 이름:홍길동
잔액:20000

계좌번호:777-88-999999
예금주 이름:김선달
잔액:30000

계좌번호:000-00-000000
예금주 이름:김미영
잔액:0

상속과 인터페이스

다형성과 메소드 오버라이딩

- [예제 6-23] 클래스 변수의 다형성을 활용하는 프로그램 (2)

```
1 class InheritanceExample8 {
2     public static void main(String args[]) {
3         EmailSender obj1 = new EmailSender("생일을 축하합니다",
4             "고객센터",
5             "admin@dukeeshop.co.kr", "10% 할인쿠폰이 발행되었습니다.");
6         SMSSender obj2 = new SMSSender("생일을 축하합니다", "고객센터",
7             "02-000-0000", "10% 할인쿠폰이 발행되었습니다.");
8         send(obj1, "hatman@yeyeye.com");
9         send(obj1, "stickman@hahaha.com");
10        send(obj2, "010-000-0000");
11    }
12    static void send(MessageSender obj, String recipient) {
13        obj.sendMessage(recipient);
14    }
15 }
```

서브클래스 객체 obj1, obj2를 가지고 메소드를 호출합니다.

슈퍼클래스 타입의 파라미터 변수

어느 클래스의 sendMessage 메소드가 호출될까요?

명령 프롬프트

```
E:\work\chap6\6-1-6\example2>java Inhe
```

제목: 생일을 축하합니다
보내는 사람: 고객센터 admin@dukeeshop.
받는 사람: hatman@yeyeye.com
내용: 10% 할인쿠폰이 발행되었습니다.

제목: 생일을 축하합니다
보내는 사람: 고객센터 admin@dukeeshop.
받는 사람: stickman@hahaha.com
내용: 10% 할인쿠폰이 발행되었습니다.

제목: 생일을 축하합니다
보내는 사람: 고객센터
전화번호: 010-000-0000
회신 전화번호: 02-000-0000
메시지 내용: 10% 할인쿠폰이 발행되었습

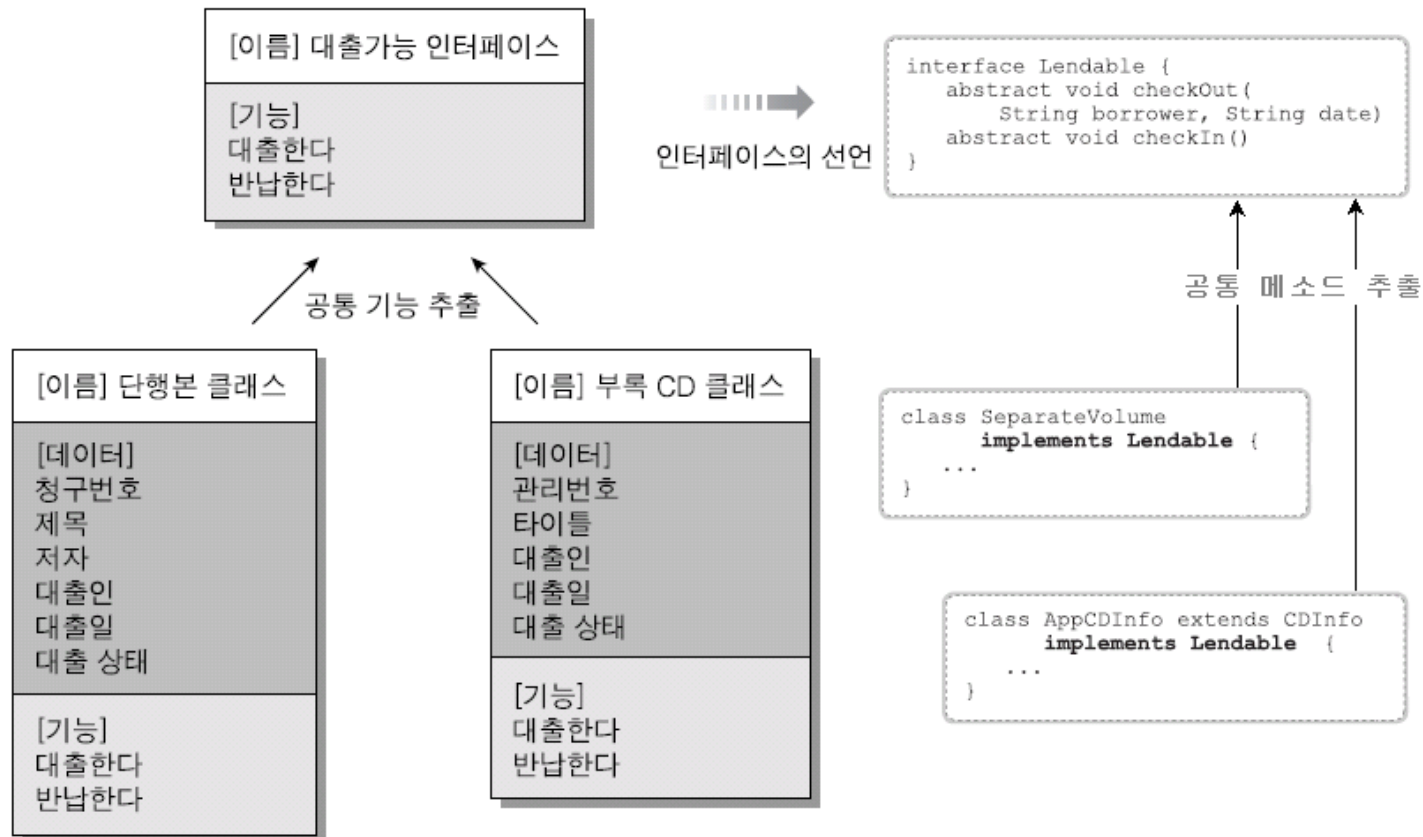
```
E:\work\chap6\6-1-6\example2>
```

상속과 인터페이스

02. 인터페이스

인터페이스란?

- 자바에서는 클래스의 다중 상속을 허용하지 않음

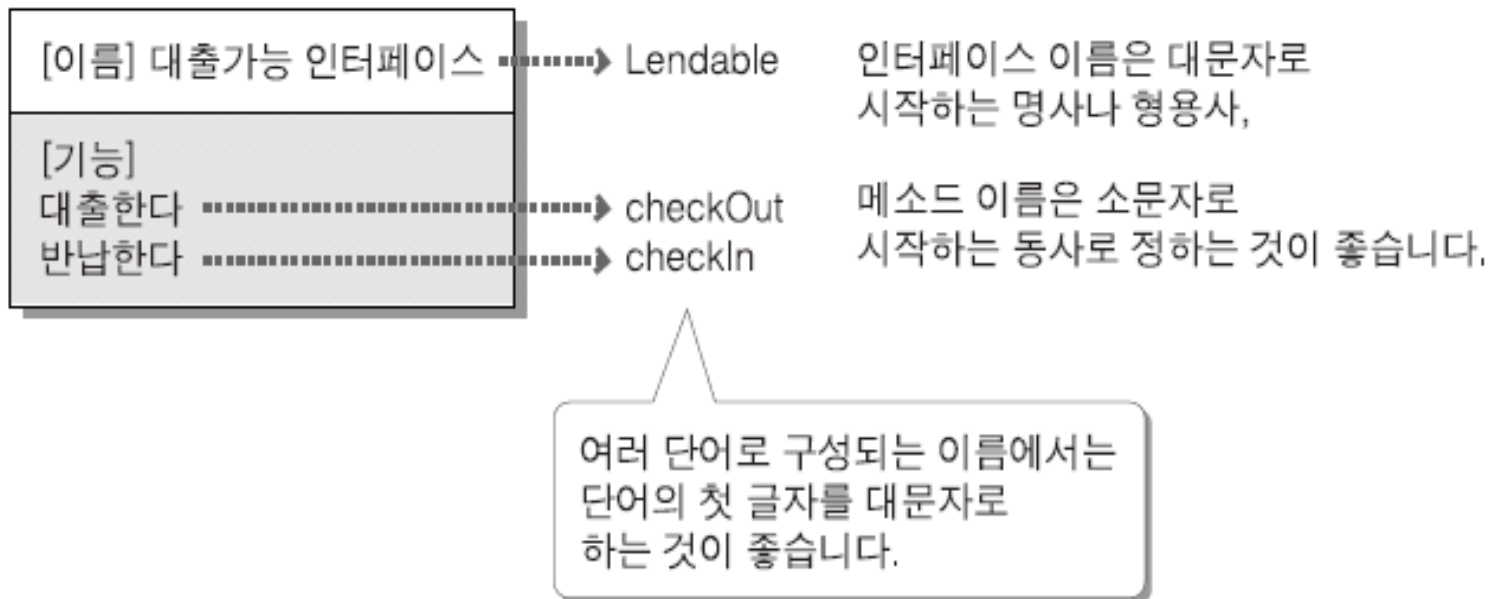


상속과 인터페이스

02. 인터페이스

인터페이스의 선언

•• 인터페이스의 선언 방법



상속과 인터페이스

02. 인터페이스

인터페이스의 선언

- [예제 6-24] 대출가능 인터페이스

```
1  interface Lendable {  
2      void checkOut(String borrower, String date);  
3      void checkIn();  
4  }
```

인터페이스의 선언을 시작하는 키워드

대출한다 메소드

반납한다 메소드

이렇게 선언해도 마찬가지로입니다.

상속과 인터페이스

02. 인터페이스

인터페이스를 구현하는 클래스의 선언

- 인터페이스를 구현하는 클래스의 선언 방법

```
class SeparateVolume implements Lendable {  
    ...  
}
```

implements 절에 반드시
써야하는 자바 키워드

인터페이스 이름

상속과 인터페이스

02. 인터페이스

인터페이스를 구현하는 클래스의 선언

- [예제 6-25] 대출가능 인터페이스를 구현하는 단행본 클래스

```
1  class SeparateVolume implements Lendable {  
2      String requestNo;    // 청구번호  
3      String bookTitle;    // 제목  
4      String writer;       // 저자  
5      String borrower;     // 대출인  
6      String checkOutDate;  // 대출일  
7      byte state;          // 대출상태  
8      SeparateVolume(String requestNo, String bookTitle, String writer) {  
9          this.requestNo = requestNo;  
10         this.bookTitle = bookTitle;  
11         this.writer = writer;  
12     }  
13     public void checkOut(String borrower, String date) { // 대출한다  
14         if (state != 0)  
15             return;  
16         this.borrower = borrower;  
17         this.checkOutDate = date;  
18         this.state = 1;  
19         System.out.println("*" + bookTitle + " 이(가) 대출되었습니다.");  
20         System.out.println("대출인:" + borrower);  
21         System.out.println("대출일:" + date + "\n");  
22     }  
23     public void checkIn() { // 반납한다  
24         this.borrower = null;  
25         this.checkOutDate = null;  
26         this.state = 0;  
27         System.out.println("*" + bookTitle + " 이(가) 반납되었습니다.\n");  
28     }  
29 }
```

implements 절

인터페이스의 메소드를 구현할 때 반드시 써야 하는 키워드

인터페이스의 메소드를 구현할 때 반드시 써야 하는 키워드

상속과 인터페이스

02. 인터페이스

인터페이스를 구현하는 클래스의 선언

- extends 절과 implements 절이 모두 있는 클래스의 선언 방법

```
class AppCDInfo extends CDInfo implements Lendable {  
    ...  
}
```

extends 절이 먼저 오고

그 다음에 implements 절이 와야 합니다.

상속과 인터페이스

02. 인터페이스

인터페이스를 구현하는 클래스의 선언

- [예제 6-26] 대출가능 인터페이스를 구현하는 부록 CD 클래스

상속

```
1 class AppCDInfo extends CDInfo implements Lendable {
2     String borrower; // 대출인
3     String checkOutDate; // 대출일
4     byte state; // 대출상태
5     AppCDInfo(String registerNo, String title) {
6         super(registerNo, title);
7     }
8     public void checkOut(String borrower, String date) {
9         if (state != 0)
10            return;
11         this.borrower = borrower;
12         this.checkOutDate = date;
13         this.state = 1;
14         System.out.println("*" + title + " CD가 대출되었습니다.");
15         System.out.println("대출인:" + borrower);
16         System.out.println("대출일:" + date + "\n");
17     }
18     public void checkIn() {
19         this.borrower = null;
20         this.checkOutDate = null;
21         this.state = 0;
22         System.out.println("*" + title + " CD가 반납되었습니다.\n");
23     }
24 }
```

```
1 class CDInfo {
2     String registerNo; // 관련번호
3     String title; // 타이틀
4     CDInfo(String registerNo, String title) {
5         this.registerNo = registerNo;
6         this.title = title;
7     }
8 }
```

상속과 인터페이스

● 인터페이스를 구현하는 클래스의 선언

- [예제 6-27] 단행본 클래스와 부록 CD 클래스를 사용하는 프로그램

```
1 class InterfaceExample1 {
2     public static void main(String args[]) {
3         SeparateVolume obj1 = new SeparateVolume("863-774개", "개미", "베르베르");
4         AppCDInfo obj2 = new AppCDInfo("2005-7001", "Redhat Fedora");
5         obj1.checkOut("김영숙", "20060315");
6         obj2.checkOut("박희경", "20060317");
7         obj1.checkIn();
8         obj2.checkIn();
9     }
10 }
```

```
E:\work\chap6\6-2-1>javac Lendable.java ----- 인터페이스의 컴파일 방법

E:\work\chap6\6-2-1>dir
E 드라이브의 볼륨: 로컬 디스크
볼륨 일련 번호: 305B-17D4

E:\work\chap6\6-2-1 디렉터리

2006-09-03  02:41p    <DIR>          .
2006-09-03  02:41p    <DIR>          ..
                                114 Lendable.java
                                1,103 SeparateVolume.java
                                215 CDInfo.java
                                882 AppCDInfo.java
                                452 InterfaceExample1.java
                                136 Dictionary.java
                                182 Lendable.class
```

```
E:\work\chap6\6-2-1>java
*개미 이<가> 대출되었습니다
대출인:김영숙
대출일:20060315

*Redhat Fedora CD가 대출되었습니다
대출인:박희경
대출일:20060317

*개미 이<가> 반납되었습니다

*Redhat Fedora CD가 반납되었습니다

E:\work\chap6\6-2-1>
```

```
E:\work\chap6\6-2-1>javac SeparateVolume.java
E:\work\chap6\6-2-1>javac CDInfo.java
E:\work\chap6\6-2-1>javac AppCDInfo.java
E:\work\chap6\6-2-1>javac InterfaceExample1.java
E:\work\chap6\6-2-1>
```

02. 인터페이스

인터페이스의 사용 방법

• 인터페이스를 가지고 할 수 없는 일

- 객체 생성에 사용하는 것은 불가능

```
obj = new Lendable(); // 잘못된 예
```

- 클래스에 메소드 로직을 상속해줄 수 없음

• 인터페이스를 가지고 할 수 있는 일

- 클래스의 선언 방법을 제한할 수 있음
- 인터페이스 변수 선언에 사용됨

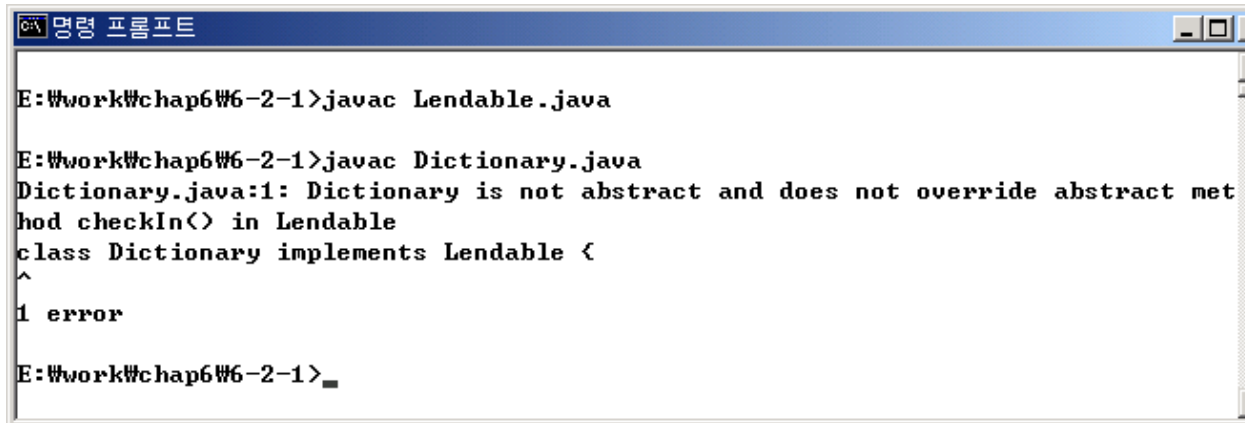
상속과 인터페이스

02. 인터페이스

클래스의 선언 방법을 제한하는 인터페이스

- [예제 6-28] Lendable 인터페이스를 구현하는 클래스의 예 – 잘못된 예

```
1    class Dictionary implements Lendable {  
2        String title;  
3        Dictionary(String title) {  
4            this.title = title;  
5        }  
6    }
```



```
E:\work\chap6\6-2-1>javac Lendable.java  
  
E:\work\chap6\6-2-1>javac Dictionary.java  
Dictionary.java:1: Dictionary is not abstract and does not override abstract met  
hod checkIn() in Lendable  
class Dictionary implements Lendable {  
^  
1 error  
  
E:\work\chap6\6-2-1>_
```

02. 인터페이스

인터페이스 변수의 다형성

- 인터페이스 변수의 선언

```
Lendable obj;
```

- 인터페이스 변수의 사용

- - 인터페이스 변수에는 그 인터페이스를 구현하는 클래스의 객체를 대입할 수 있음

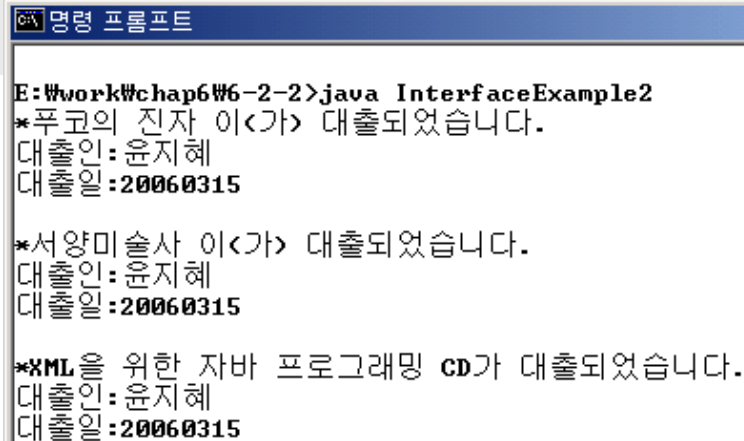
```
obj = new SeparateVolume("863-774개", "개미", "베르베르");  
obj = new AppCDInfo("2006-7001", "Redhat Fedora");
```


상속과 인터페이스

● 인터페이스 변수의 다형성

- [예제 6-29] Lendable 인터페이스 변수의 다형성을 이용하는 프로그램

```
1  class InterfaceExample2 {
2      public static void main(String args[]) {
3          Lendable arr[] = new Lendable[3]; ----- 인터페이스 타입의 배열
4          arr[0] = new SeparateVolume("8830", "푸코의 진자", "에코");
5          arr[1] = new SeparateVolume("609.2", "서양미술사", "곰브리치"); } ----- 배열에 여러 타입의 객체 저장
6          arr[2] = new AppCDInfo("02-17", "XML을 위한 자바 프로그래밍");
7          checkOutAll(arr, "윤지혜", "20060315");
8      } ----- 배열을 파라미터로 넘겨줍니다
9      static void checkOutAll(Lendable arr[], String borrower, String date) {
10         for (int cnt = 0; cnt < arr.length; cnt++)
11             arr[cnt].checkOut(borrower, date); ----- 배열의 모든 항목에 대해
12         } ----- checkOut 메소드 호출
13     }
```



```
E:\work\chap6\6-2-2>java InterfaceExample2
*푸코의 진자 이<가> 대출되었습니다.
대출인:윤지혜
대출일:20060315

*서양미술사 이<가> 대출되었습니다.
대출인:윤지혜
대출일:20060315

*XML을 위한 자바 프로그래밍 CD가 대출되었습니다.
대출인:윤지혜
대출일:20060315
```

02. 인터페이스

인터페이스의 상수 필드

- 인터페이스의 상수 필드 선언 방법 (1)

```
final static int MAXIMUM = 100;
```

- 인터페이스의 상수 필드 선언 방법 (2)

```
static int MAXIMUM = 100;  
final int MAXIMUM = 100;  
int MAXIMUM = 100;
```

상속과 인터페이스

02. 인터페이스

인터페이스의 상수 필드

- [예제 6-30] 상수 필드가 추가된 대출가능 인터페이스

```
1  interface Lendable {  
2      final static byte STATE_BORROWED = 1;    // 대출 중  
3      final static byte STATE_NORMAL = 0;      // 대출되지 않은 상태  
4      void checkOut(String borrower, String date);  
5      void checkIn();  
6  }
```

상속과 인터페이스

● 인터페이스의 상수 필드

- [예제 6-31] 대출가능 인터페이스로부터 상수 필드를 상속받는 단행본 클래스

```
1  class SeparateVolume implements Lendable {
2      String requestNo; // 청구번호
3      String bookTitle; // 제목
4      String writer; // 저자
5      String borrower; // 대출인
6      String checkOutDate; // 대출일
7      byte state; // 대출상태
8      SeparateVolume(String requestNo, String bookTitle, String writer) {
9          this.requestNo = requestNo;
10         this.bookTitle = bookTitle;
11         this.writer = writer;
12     }
13     public void checkOut(String borrower, String date) {
14         if (state != STATE_NORMAL)
15             return;
16         this.borrower = borrower;
17         this.checkOutDate = date;
18         this.state = STATE_BORROWED;
19         System.out.println("*" + bookTitle + " 이(가) 대출되었습니다.");
20         System.out.println("대출인:" + borrower);
21         System.out.println("대출일:" + date + "\n");
22     }
23     public void checkIn() {
24         this.borrower = null;
25         this.checkOutDate = null;
26         this.state = STATE_NORMAL;
27         System.out.println("*" + bookTitle + " 이(가) 반납되었습니다.\n");
28     }
29 }
```

implements 절

Lendable 인터페이스의 상수 필드를 사용

Lendable 인터페이스의 상수 필드를 사용

Lendable 인터페이스의 상수 필드를 사용

상속과 인터페이스

● 인터페이스의 상수 필드

- [예제 6-32] Lendable 인터페이스의 상수 필드를 사용하는 프로그램

```
1  class InterfaceExample3 {
2      public static void main(String args[]) {
3          SeparateVolume obj = new SeparateVolume("863ㄴ", "나무", "베르베르");
4          printState(obj);
5          obj.checkOut("이수경", "20060317");
6          printState(obj);
7      }
8      static void printState(SeparateVolume obj) {
9          if (obj.state == Lendable.STATE_NORMAL) { ----- Lendable 인터페이스의 상수 필드를 사용
10             System.out.println("-----");
11             System.out.println("대출상태: 대출가능");
12             System.out.println("-----");
13         }
14         if (obj.state == Lendable.STATE_BORROWED) { ----- Lendable 인터페이스의 상수 필드를 사용
15             System.out.println("-----");
16             System.out.println("대출상태: 대출중");
17             System.out.println("대출인: " + obj.borrower);
18             System.out.println("대출일: " + obj.checkOutDate);
19             System.out.println("-----");
20         }
21     }
22 }
```

명령 프롬프트

E:\work\chap6\6-2-3>java InterfaceExample3

대출상태: 대출가능

*나무 이<가> 대출되었습니다.

대출인: 이수경

대출일: 20060317

대출상태: 대출중

대출인: 이수경

대출일: 20060317

E:\work\chap6\6-2-3>_

상속과 인터페이스

익셉션을 발생하는 추상 메소드

- [예제 6-33] `checkOut` 메소드 밖으로 익셉션을 던지는 단행본 클래스

```
1 class SeparateVolume implements Lendable {
2     String requestNo;
3     String bookTitle;
4     String writer;
5     String borrower;
6     String checkOutDate;
7     byte state;
8     SeparateVolume(String requestNo, String bookTitle, String writer) {
9         this.requestNo = requestNo;
10        this.bookTitle = bookTitle;
11        this.writer = writer;
12    }
13    public void checkOut(String borrower, String date) throws Exception {
14        if (state != 0)
15            throw new Exception("*대출불가:" + bookTitle);
16        this.borrower = borrower;
17        this.checkOutDate = date;
18        this.state = 1;
19        System.out.println("*" + bookTitle + " 이(가) 대출되었습니다.");
20        System.out.println("대출인:" + borrower);
21        System.out.println("대출일:" + date + "\n");
22    }
23    public void checkIn() {
24        this.borrower = null;
25        this.checkOutDate = null;
26        this.state = 0;
27        System.out.println("*" + bookTitle + " 이(가) 반납되었습니다.\n");
28    }
29 }
```

throws 절

throws Exception

대출 중 상태(0)이면 메소드 밖으로 익셉션을 던집니다.

명령 프롬프트

E:\work\chap6\6-2-4>javac Lendable.java ----- [예제 6-24]

```
E:\work\chap6\6-2-4>javac SeparateVolume.java
SeparateVolume.java:13: checkOut<java.lang.String,java.lang.
volume cannot implement checkOut<java.lang.String,java.lang.
overridden method does not throw java.lang.Exception
    public void checkOut(String borrower, String date)
                ^
```

1 error

E:\work\chap6\6-2-4>

상속과 인터페이스

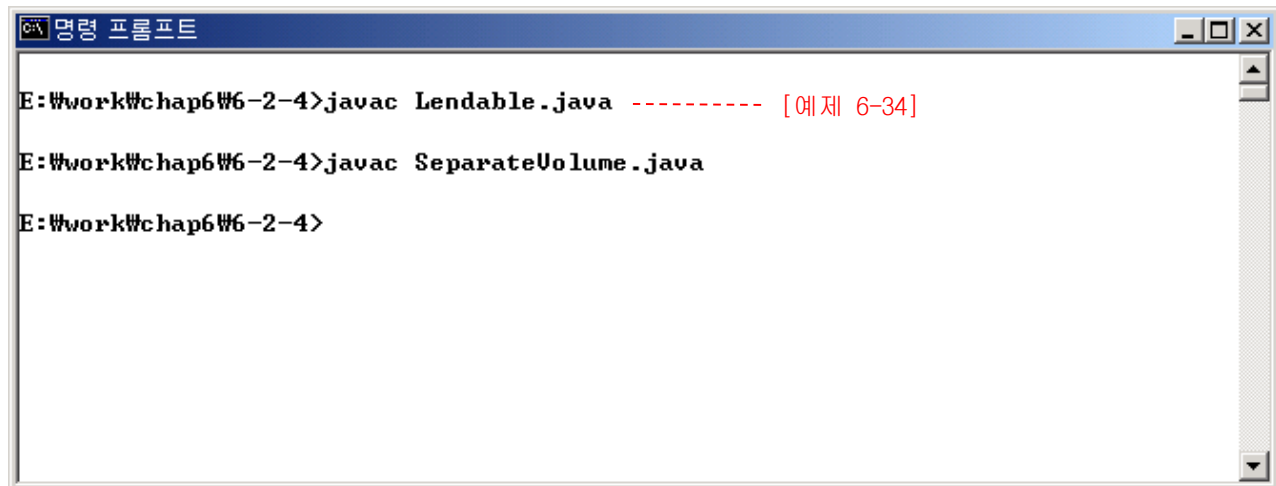
02. 인터페이스

익셉션을 발생하는 추상 메소드

- [예제 6-34] 추상 메소드에 throws 절을 추가한 대출가능 인터페이스

```
1 interface Lendable {  
2     abstract void checkOut(String borrower, String date) throws Exception;  
3     abstract void checkIn();  
4 }
```

throws 절



```
명령 프롬프트  
E:\work\chap6\6-2-4>javac Lendable.java ----- [예제 6-34]  
E:\work\chap6\6-2-4>javac SeparateVolume.java  
E:\work\chap6\6-2-4>
```

상속과 인터페이스

02. 인터페이스

인터페이스의 상속

- 인터페이스의 상속이 필요한 경우의 예

[이름] 위치이동 인터페이스

[기능]

절대위치로 이동한다

상대위치만큼 이동한다

[이름] 변환 인터페이스

[기능]

절대위치로 이동한다

상대위치만큼 이동한다

크기를 변경한다

상속과 인터페이스

02. 인터페이스

인터페이스의 상속

- 다른 인터페이스를 상속 받는 인터페이스의 선언 방법

```
interface Transformable extends Movable {  
    ...  
}
```

자바 키워드

상속할 인터페이스 이름

상속과 인터페이스

02. 인터페이스

인터페이스의 상속

- [예제 6-35] 위치이동 인터페이스를 상속하는 변환 인터페이스

위치이동 인터페이스

```
1  interface Movable {  
2      void moveTo(int x, int y);           // 절대 위치로 이동한다  
3      void moveBy(int xOffset, int yOffset); // 상대 위치만큼 이동한다  
4  }
```

상속

변환 인터페이스

```
1  interface Transformable extends Movable {  
2      void resize(int width, int height); // 크기를 변경한다  
3  }
```

상속과 인터페이스

02. 인터페이스

인터페이스의 상속

- [예제 6-36] 변환 인터페이스를 구현하는 사각형 클래스

```
1  class Rectangle implements Transformable {
2      int x, y, width, height;
3      Rectangle(int x, int y, int width, int height) {
4          this.x = x;
5          this.y = y;
6          this.width = width;
7          this.height = height;
8      }
9      public void resize(int width, int height) {
10         this.width = width;
11         this.height = height;
12     }
13     public void moveTo(int x, int y) {
14         this.x = x;
15         this.y = y;
16     }
17     public void moveBy(int xoffset, int yoffset) {
18         this.x += xoffset;
19         this.y += yoffset;
20     }
21 }
```

Transformable 인터페이스의 메소드를 구현합니다.

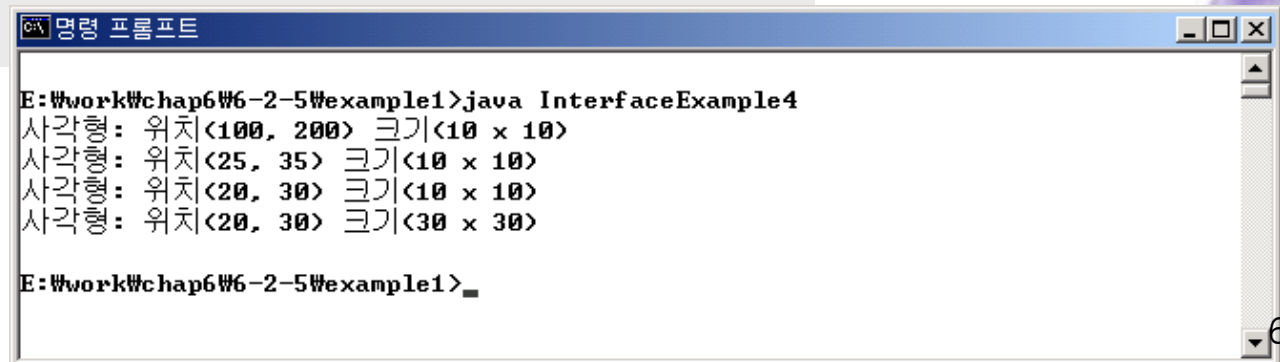
Movable 인터페이스의 메소드를 구현합니다.

상속과 인터페이스

● 인터페이스의 상속

- [예제 6-37] Rectangle 클래스를 사용하는 프로그램

```
1  class InterfaceExample4 {  
2      public static void main(String args[]) {  
3          Rectangle obj = new Rectangle(100, 200, 10, 10);  
4          printRectangle(obj);  
5          obj.moveTo(25, 35);  
6          printRectangle(obj);  
7          obj.moveBy(-5, -5);  
8          printRectangle(obj);  
9          obj.resize(30, 30);  
10         printRectangle(obj);  
11     }  
12     static void printRectangle(Rectangle obj) {  
13         System.out.printf("사각형: 위치(%d, %d) 크기(%d x %d) %n",  
14                             obj.x, obj.y, obj.width, obj.height);  
15     }  
}
```



```
E:\work\chap6\6-2-5\example1>java InterfaceExample4  
사각형: 위치<100, 200> 크기<10 x 10>  
사각형: 위치<25, 35> 크기<10 x 10>  
사각형: 위치<20, 30> 크기<10 x 10>  
사각형: 위치<20, 30> 크기<30 x 30>  
  
E:\work\chap6\6-2-5\example1>
```

상속과 인터페이스

02. 인터페이스

인터페이스의 다중 상속

•• 인터페이스의 다중 상속이 필요한 경우의 예

[이름] 크기변경 인터페이스

[기능]
크기를 변경한다

[이름] 색상변경 인터페이스

[기능]
전경색을 바꾼다
배경색을 바꾼다

[이름] 외형변경 인터페이스

[기능]
크기를 변경한다
전경색을 바꾼다
배경색을 바꾼다
폰트를 바꾼다

상속과 인터페이스

02. 인터페이스

인터페이스의 다중 상속

- 다중 상속을 하는 인터페이스의 선언 방법

```
interface Changeable extends Resizable, Colorable {  
    ...  
}
```

자바 키워드

상속할 인터페이스 이름

상속과 인터페이스

02. 인터페이스

인터페이스의 다중 상속

- [예제 6-38] 두 개의 인터페이스를 동시에 상속받는 외형변경 인터페이스

크기변경 인터페이스 선언

```
1 interface Resizable {  
2     void resize(int width, int  
3     height);  
4 }
```

색상변경 인터페이스 선언

```
1 interface Colorable {  
2     void setForeground(String color);  
3     void setBackground(String color);  
4 }
```

외형변경 인터페이스 선언

```
1 interface Changeable extends Resizable, Colorable {  
2     void setFont(String font);  
3 }
```

상속과 인터페이스

02. 인터페이스

인터페이스의 다중 상속

- [예제 6-39] 외형변경 인터페이스를 구현하는 라벨 클래스

```
1  class Label implements Changeable {
2      String text;
3      int width, height;
4      String foreground, background;
5      String font;
6      Label(String text, int width, int height, String foreground, String background, String font) {
7          this.text = text;
8          this.width = width;
9          this.height = height;
10         this.foreground = foreground;
11         this.background = background;
12         this.font = font;
13     }
14     public void resize(int width, int height) {
15         this.width = width;
16         this.height = height;
17     }
18     public void setForeground(String color) {
19         this.foreground = color;
20     }
21     public void setBackground(String color) {
22         this.background = color;
23     }
24     public void setFont(String font) {
25         this.font = font;
26     }
27 }
```

Resizable 인터페이스의 메소드를 구현

Colorable 인터페이스의 메소드를 구현

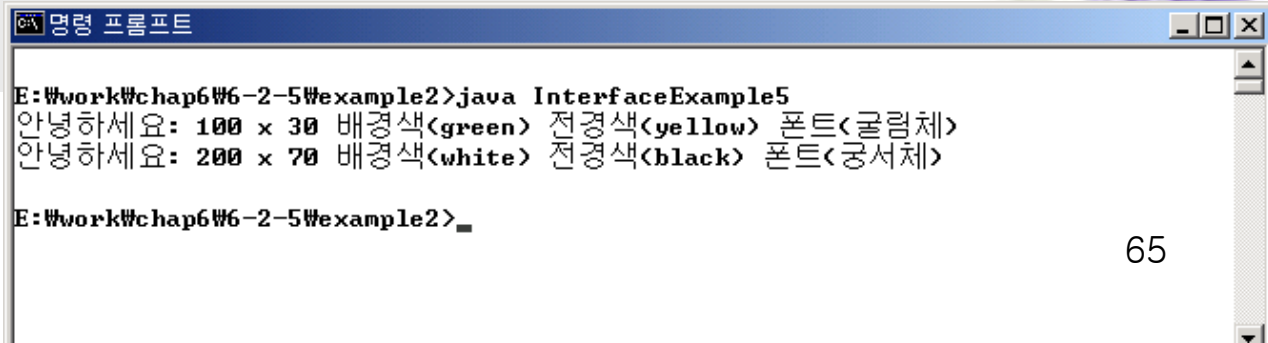
Changeable 인터페이스의 메소드를 구현

상속과 인터페이스

● 인터페이스의 다중 상속

- [예제 6-40] 라벨 클래스를 사용하는 프로그램

```
1  class InterfaceExample5 {
2      public static void main(String args[]) {
3          Label obj = new Label("안녕하세요", 100, 30, "yellow", "green", "굴림체");
4          printLabel(obj);
5          obj.resize(200, 70);
6          obj.setForeground("black");
7          obj.setBackground("white");
8          obj.setFont("궁서체");
9          printLabel(obj);
10     }
11     static void printLabel(Label obj) {
12         System.out.printf(
13             "%s: %d x %d 배경색(%s) 전경색(%s) 폰트(%s) %n",
14             obj.text, obj.width, obj.height,
15             obj.background, obj.foreground, obj.font);
16     }
17 }
```



```
명령 프롬프트
E:\work\chap6\6-2-5\example2>java InterfaceExample5
안녕하세요: 100 x 30 배경색(green) 전경색(yellow) 폰트(굴림체)
안녕하세요: 200 x 70 배경색(white) 전경색(black) 폰트(궁서체)

E:\work\chap6\6-2-5\example2>
```

수 고 하 셜 습 니다!