실습문제 5-8

1

- 교재 318페이지, 실습문제 8
- 프로젝트 이름: prac05_08
- 클래스 이름: PointApp_학번
  - Point, PositivePoint, PointApp_학번 클래스를 작성
  - PointApp_학번 클래스에는 main() 함수만 존재
    - main 함수를 PositivePoint에 넣지 말고 이 클래스에 넣을 것
  - 클래스 PositivePoint는 클래스 Point를 상속 받아서 사용
  - 클래스 PositivePoint, Point에는 move 메소드가 존재
    - 실질적으로 좌표값을 움직이는 메소드는 클래스 Point의 move 메소드
  - 클래스 PositivePoint의 move 메소드가 하는 일
    - 좌표값 양수 이면 클래스 Point의 move 함수 호출
    - 그렇지 않은 경우에는 하는 것이 없음

실습문제 5-8 소스: 프로그램 전체 윤곽

2

```
class Point {
    private int x, y;
    public Point(int x, int y) { this.x = x; this.y = y; }
    public int getX() { return x; }
    public int getY() { return y; }
    protected void move(int x, int y) { this.x = x; this.y = y; }
}
class PositivePoint 부모_클래스인_Point_클래스_상속 {      }

public class PointApp_학번 {
    public static void main(String[] args) {
        PositivePoint p = new PositivePoint();
        p.move(10, 10);
        System.out.println(p.toString() + "입니다.");

        p.move(-5, 5); // 객체 p는 음수 공간으로 이동되지 않음
        System.out.println(p.toString() + "입니다.");

        PositivePoint p2 = new PositivePoint(-10, -10);
        System.out.println(p2.toString() + "입니다.");
    }
}
```

실습문제 5-8 소스

3

```
// 프로그램의 전체 윤곽

class Point {
   // 교재 316페이지 하단의 Point 클래스를 보고 입력한다.
}

class PositivePoint 부모_클래스인_Point_클래스_상속 {
   클래스 생성자() {                                  // 뒷 페이지 참조
   }
   클래스 생성자(int x, int y) {                      // 뒷 페이지 참조
   }
   부모 클래스의 move 함수 오버라이딩 {               // 뒷 페이지 참조
   }
   public String toString() {                         // 뒷 페이지 참조
   }
}

public class PointApp_학번 {
   // 교재 318페이지, 실습문제 8번의 main() 함수를 여기에 삽입
}
```

실습문제 5-8 소스

4

```
   클래스 생성자() {
      x, y 좌표 값이 (0, 0)이 되게 부모 클래스 생성자를 호출함; // super 키워드 사용
   }

   클래스 생성자(int x, int y) {
      super 키워드를 사용하여 부모 클래스 생성자를 호출함;
      if (x, y가 둘 중 하나라도 0보다 작으면)  // 연산자 || 사용
         부모 클래스의 move함수 호출(super 이용)하여 (0,0)로 이동시킴;
   }

   부모 클래스의 move 함수 오버라이딩 {
      x, y 둘 다 0보다 크면 부모 클래스의 move 함수 호출; // if문 사용
      아니면 그냥 리턴
   }

   public String toString() {
      return "(" + ____ + "," + ____ + ")의 점"; // x 값과 y 값을 얻는 함수를 호출
   }
```

실습문제 5-10

5

- 교재 319페이지, 실습문제 10
- 프로젝트 이름: prac05_10
- 클래스 이름: DictionaryApp_학번
  - Dictionary 생성자 : 키와 값의 쌍의 최대 개수를 입력 받음

```
public Dictionary(int capacity) { // 생성자
    keyArray = new String [capacity];
    valueArray = new String [capacity];
}
```

  - String 데이터의 비교
    - 클래스 String의 equals() 메소드 사용
  - 입력된 키의 개수를 저장하고 있는 변수 count 필요
  - 인스턴스를 생성하려면 모든 추상 메소드가 구현 되어야 함

실습문제 5-10 소스: 교재 내용

6

```
abstract class PairMap {
    protected String keyArray [];                   // key들을 저장하는 배열
    protected String valueArray [];                 // value 들을 저장하는 배열
    abstract String get(String key);                // key 값으로 value를 검색
    abstract void put(String key, String value);    // key와 value를 쌍으로 저장
    abstract String delete(String key);   // key 값을 가진 아이템(value와 함께)을 삭제.
                                                    // 삭제된 value 값 리턴
    abstract int length();                          // 현재 저장된 아이템의 개수 리턴
}
class Dictionary 부모_클래스인_ PairMap_클래스_상속 { 뒷 페이지 참조 }
클래스 DictionaryApp_학번 {
    public static void main(String[] args) {
        Dictionary dic = new Dictionary(10);
        dic.put("황기태", "자바");
        dic.put("이재문", "파이선");
        dic.put("이재문", "C++");          // 이재문의 값을 C++로 수정
        System.out.println("이재문의 값은 " + dic.get("이재문"));
        System.out.println("황기태의 값은 " + dic.get("황기태"));
        dic.delete("황기태");
        System.out.println("황기태의 값은 " + dic.get("황기태"));
    }
}
```

실습문제 5-10 소스: 프로그램 전체 윤곽

7

```
// main이 포함된 클래스만 public으로 선언하고 나머지 클래스는
// 디폴트 클래스로 선언, 즉 public 선언하지 말 것
abstract class PairMap {
   // 교재 319페이지, 실습문제 10을 보고 입력한다.
}
class Dictionary 부모_클래스인_ PairMap_클래스_상속 {
   // count 변수 선언 및 초기화(0으로);
   // count는 외부에서 접근 못하며, 배열에 저장된 키의 개수 값을 가질 예정임
   클래스 생성자() {                                  // 앞의 앞 페이지 참조
   }
   부모 클래스의 get 함수 오버라이딩 {                 // 뒷 페이지 참조
   }
   부모 클래스의 put 함수 오버라이딩 {                 // 뒷 페이지 참조
   }
   부모 클래스의 delete 함수 오버라이딩 {              // 뒷 페이지 참조
   }
   부모 클래스의 length 함수 오버라이딩 {              // 뒷 페이지 참조
   }
}
DictionaryApp_학번 클래스 {
   // 교재 320페이지, 실습문제 10번 main() 함수를 여기에 삽입
}
```

실습문제 5-10 소스

8

```
   부모 클래스의 get 함수 오버라이딩 {
      for (int i = 0 현재 배열에 저장된 키의 개수(count)만큼 반복) { // 배열 길이만큼이 아님
         keyArray[i] 문자열과 함수 인자인 key문자열이 같으면, // if 사용
            valueArray[i]를 리턴함; // 참고로 'a와 b 문자열이 같으면'은 if (a.equals(b)) 로 호출함
      } // 배열에 key와 같은 문자열이 있을 경우 for문 안에서 이미 리턴했음
      배열에 key와 같은 문자열이 없는 경우에 null 리턴함;
   }
   부모 클래스의 put 함수 오버라이딩 {
      int i;
      for (i = 0 현재 배열에 저장된 키의 개수(count) 만큼 반복) {
         keyArray[i] 문자열과 함수 인자인 key문자열이 같으면, for문을 탈출
      }
      키 값을 발견하지 못했다면 { // for문을 끝까지 다 돌았다면 i 값은 무슨 변수 값과 같을까?
         i가 keyArray 배열 크기(길이)보다 작다면 {   // 배열이 꽉 차지 않은 경우
            key를 keyArray[]의 기존 저장된 맨 뒤에 추가하고, value를 valueArray[]의 맨 뒤에 추가함;
            // 이 때 배열 keyArray[]와 valueArray[]의 맨 뒤의 위치는? (count의 위치)
            저장된 키 값이 하나 추가 되었으므로 count(배열에 저장된 키의 개수) 값을 증가;
         } // else 즉, 이미 배열이 찼다면 더 이상 삽입할 수 없으므로 아무 것도 하지 않음
      }
      키 값을 발견했다면 { // valueArray[]의 기존 원소 값을 수정
         키 값을 발견한 i 위치의 배열 valueArray의 값을 value로 수정, count는 증가시키지 않음
      }
   }
```

실습문제 5-10 소스

9

```
부모 클래스의 delete 함수 오버라이딩 {
   int i;
   for (i = 0 현재 배열에 저장된 키의 개수만큼 반복) {
       keyArray[i] 문자열과 함수 인자인 key문자열이 같으면, for문을 탈출
   }
   키 값을 발견하지 못했다면, null 값을 리턴;
   String클래스 변수 value를 선언하고 찾은 키 값과 같은 위치에 있는 valueArray[] 값으로 초기화;
   // 찾은 키 값과 value 값을 삭제; 이 경우 그 뒤쪽의 값을 모두 앞으로 옮겨야 함
   for (int j = 키 값을 찾은 위치에서 (마지막-1)까지) {
      keyArray[j+1]의 값을 keyArray[j]에 저장; // 뒤 쪽의 값을 앞쪽으로 옮김
      valueArray[j+1]의 값을 valueArray[j]에 저장; // [j+1] 때문에 (마지막-1)까지 해야 함
   }
   저장된 키 값이 하나 삭제 되었으므로 count를 변경해야 함;
   value 변수 값을 리턴; // 삭제된 key에 상응하는 value 값임
}

부모 클래스의 length 함수 오버라이딩 {
   count 값을 리턴;
}
```