EXERCISE
INTRODUCTION TO JAVA PROGRAMMING

1. 다음의 질문에 간단히 답하시오.

❶ int형을 저장하는 ArrayList를 생성할 수 있는가?

❷ Iterator 인터페이스는 어떤 목적으로 사용되는가?

❸ 리스트(list)와 집합(set)의 차이점은 무엇인가?

❹ TreeSet, HashSet, LinkedListSet의 차이점은 무엇인가?

❺ 키와 값의 매핑을 나타내는 인터페이스는 무엇인가?

2. 다음 문장이 참인지 거짓인지를 말하고 거짓이면 올바르게 수정하라.

❶ 리스트 안에는 원소들이 중복될 수 있다.

❷ 집합 안에는 원소들이 중복될 수 있다.

❸ ListIterator는 리스트 안에서 역방향으로 움직일 수 있다.

❹ LinkedList에서는 원소들을 랜덤하게 접근할 수 있다.

❺ HashSet에서는 원소들이 추가된 순서대로 추출된다.

3. 다음은 Stack 클래스의 일부분이다.

❶ Stack에 저장되는 데이터의 타입을 int 대신에 제네릭 타입으로 표시하여 보자.

```
public class Stack {
   private int[] stack;
   public void push(int data) { .... }
   public int pop() { .... }
}
```

❷ String 타입의 데이터를 가지는 Stack을 생성하는 문장을 쓰시오.

4. 다음과 같은 ArrayList를 제네릭을 이용하여 생성하여 보자.

❶ 20개의 String을 저장할 수 있는 ArrayList

❷ 30개의 Double을 저장할 수 있는 ArrayList

5. 다음과 같이 리스트가 생성되었다고 하자. 다음의 각 문장을 실행한 후의 결과를 쓰시오.

```
String[] s = { "사과", "배", "바나나" };
ArrayList list = new ArrayList(Arrays.asList(s));
```

❶ `list.add("포도"); System.out.println(list);`

❷ `list.add(2, "자몽"); System.out.println(list);`

❸ `System.out.println(list.get(3));`

❹ `list.remove(1);System.out.println(list);`

❺ `System.out.println(list.contains("사과"));`

❻ `System.out.println(list.indexOf("사과"));`

6. 다음과 같이 집합(set)이 생성되었다고 하자. 다음의 각 문장을 실행한 후의 결과를 쓰시오.

```
String[] s1 = { "사과", "배", "바나나" };
String[] s2 = { "사과", "귤", "수박" };
HashSet set1 = new HashSet(Arrays.asList(s1));
HashSet set2 = new HashSet(Arrays.asList(s2));
```

❶ `set1.add("메론"); System.out.println(set1);`

❷ `set1.remove("배"); System.out.println(set1);`

❸ `System.out.println(set1.contains("귤"));`

❹ `System.out.println(set1.containsAll(set2));`

❺
```
Set intersection = new HashSet(set1);
intersection.retainAll(set2); System.out.println(intersection);
```

❻
```
Set union = new HashSet(set1);
union.addAll(set2); System.out.println(union);
```

7. `list`가 `ArrayList<Double>`의 객체를 참조하고 있다고 하자. `list`의 모든 원소를 출력하는 문장을 다음과 같이 작성하라.

❶ 인덱스 변수를 사용하는 보통의 for 루프

❷ for-each 구문을 사용

❸ Iterator를 사용

PROGRAMMING
INTRODUCTION TO JAVA PROGRAMMING

1. 랜덤 리스트(random list)를 작성하여 보자. 랜덤 리스트란 원소들을 가지고 있다가 get()이라는 메소드를 호출하면 랜덤하게 하나의 원소를 선택하여서 반환한다. 모든 타입의 객체를 저장하도록 제네릭을 사용하라. 원소들은 ArrayList를 이용하여서 저장하고 add() 메소드는 원소를 추가한다. select()가 호출되면 난수 생성기를 이용하서 ArrayList 원소 중에서 하나를 선택하여 반환하라. 다음 코드를 참조하라.

```
public class RandomList<T> {
   _____________________________________;
   public void add(T  item) { ____________________;}
   public T select() { ____________________________;}
}
```

2. 타입 매개 변수 T를 가지는 클래스 MyMath를 작성하여 보자. MyMath에는 최소값을 구하는 getMinimun() 메소드를 추가하여 보자. Integer나 Double과 같은 다양한 타입의 데이터에 대하여 최소값을 구할 수 있도록 하라.

3. 제네릭을 사용하여서 똑같은 타입의 객체 두 개를 저장하는 Pair 클래스를 작성하여 보자. 생성자와 접근자, 설정자, toString() 메소드를 정의하라. String을 저장하여 다음과 같이 테스트하여 보라.

```
MyPair<String> fruit = new MyPair<String>("사과", "포도");
```

4. 클래스 안에서 하나의 메소드만 제네릭으로 만들어보자. 제네릭 메소드 a()를 가지는 클래스 Test를 정의하여 보자. 메소드 a()는 매개 변수의 클래스 이름을 출력한다. 객체 obj의 클래스 이름은 obj.getClass().getName()으로 출력할 수 있다. int나 float와 같은 기초형 값을 전달하여서 호출해보자. 어떤 값이 출력되는가?

5. 자바에서도 스택(stack)이 제공되지만 우리 나름대로의 스택을 작성하여 보자. 스택이란 먼저 들어간 데이터가 나중에 나오는 자료 구조이다. 스택은 1차원 배열과

top이라는 정수 변수를 이용하여서 간단히 구현할 수 있다. 타입 매개 변수를 사용하여서 어떤 타입도 저장할 수 있는 스택을 설계하여 보라.

6. 큐(queue)는 먼저 들어간 데이터가 먼저 나오는 자료 구조이다. 큐는 `LinkedList`를 이용하면 간단하게 구현할 수 있다. 클래스의 이름을 `LinkedListQueue`라고 하고 큐에 데이터를 추가하는 메소드를 `enqueue()`라고 하고 큐에서 데이터를 삭제하는 메소드를 `dequeue()`로 구현하라. `enqueue()`는 `LinkedList`의 `addLast()` 메소드를 이용하면 쉽게 구현할 수 있고 `dequeue()`는 `removeFirst()`를 이용하면 쉽게 구현이 가능하다. 타입 매개 변수 T를 사용하여서 어떤 타입의 데이터도 저장할 수 있도록 구현하라.

7. 장기 자랑 프로그램에 사용될 수 있는 심사 위원들의 점수를 집계하는 프로그램을 작성하라. 점수는 0.0에서 10.0까지 가능하다. 10명의 점수 중에서 최저 점수와 최고 점수는 제외된다. `Double` 타입의 `ArrayList`를 사용하라.

8. 학생들의 정보를 `ArrayList`에 저장하고 검색할 수 있는 프로그램을 작성하라. 학생들의 정보는 `Student`라는 클래스로 나타낸다. `Student`는 학생의 이름, 주소, 전화번호 등의 필드로 가진다. 적절한 접근자와 설정자를 작성하라. 학생들의 정보를 추가하고 검색하고 삭제하는 간단한 메뉴를 제공한다. `ArrayList`의 원소들을 처리할 때 `for-each` 루프를 사용하라.

9. 소수를 구하는데 고대의 그리스 수학자 에라스토스테네스에 의하여 개발된 에라스토스테네스의 체(Sieve of Erastosthenes)라는 알고리즘이 있다. 이 알고리즘은 정해진 범위안의 소수를 찾아주는 비교적 간단한 방법이다. 처음에는 리스트 안의 모든 정수가 소수라고 가정한다. 알고리즘은 가장 작은 소수인 2부터 시작한다. 상식적으로 2의 배수는 소수가 아닌 것이 확실하다. 정해진 범위 안에서 2의 배수를 모두 찾아서 리스트에서 제거한다. 즉 2, 4, 6, 8, ..정수는 소수에서 제외된다. 다시 3의 배수(3, 6, 9, ...)를 찾아서 리스트에서 제거한다. 정해진 범위에 대하여 이와 같은 절차를 완료한 후에도 리스트에 남아 있는 원소가 바로 소수가 된다. 이 알고리즘을 이용하여 2부터 100사이의 소수를 찾아보라. 2부터 100까지의 값으로 초기화된 `LinkedList`를 사용하라. 리스트 안에 남아 있는 정수를 스캔하는데 `Iterator`를 사용하라.

10. 로또 번호를 생성하는 프로그램을 작성하여 보자. 로또는 1부터 45까지의 숫자 중에서 6개를 선택한다. 로또 번호는 중복되면 안 된다. 따라서 집합을 나타내는 HashSet을 사용하여서 중복을 검사하여 보자. Math.random()을 사용하면 0부터 1사이의 난수를 생성할 수 있다. 0부터 1사이의 난수가 생성되면 여기에 45를 곱하고 1을 더하면 1부터 45사이의 정수를 생성할 수 있다. 생성된 정수는 HashSet의 contains() 메소드를 이용하여서 이미 선택된 정수인지를 검사한다.

11. Map을 사용하여서 영어 사전을 구현하여 보자. 사용자가 단어를 입력하면 단어의 설명을 보여준다. 사전에 단어를 추가하거나 검색, 삭제할 수 있는 간단한 메뉴 시스템을 만든다.

LAB
INTRODUCTION TO JAVA PROGRAMMING

1. 다양한 동물을 가둘 수 있는 우리를 설계하여 보자. 클래스 이름은 Cage로 하자.

```
class Animal{       }
class Lion extends Animal {          }
class Tiger extends Animal {          }

public class Cage {
   private Object animal;
   public void setAnimal(Object x) {
      animal = x;
   }
   public Object getAnimal() {
      return animal;
   }
   public static void main(String[] args) {
         Cage cage = new Cage();
         cage.setAnimal(new Lion());
         Lion lion = cage.getAnimal();              //❶
  }
}
```

❶ 위 소스의 ❶번 문장에서 오류가 발생하는 이유는 무엇인가?

❷ ❶번 문장을 다음과 같이 변경하면 어떻게 되는가?

```
Lion lion = (Lion)cage.getAnimal();              //❶
```

❸ 만약 Lion 객체를 Cage에 넣고 꺼낼 때는 Tiger로 형변환하면 어떻게 되는가? 컴파일시 오류가 감지되는가? 실행시에 오류가 감지되는가?

```
cage.setAnimal(new Lion());
Tiger tiger = (Tiger)cage.getAnimal();           //❶
```

❹ 다양한 타입의 동물을 우리에 가둘 수 있도록 위의 클래스를 제네릭을 사용하여서 다시 작성하여 보라. 즉 Cage 안에 저장되는 타입을 T로 표시하라.

❺ Cage 클래스 안에 제네릭을 사용하는 ArrayList를 포함하여서 같은 타입의 동물을 여러 마리 가둘 수 있도록 변경하여 보자. 적절한 필드, 생성자, 메소드

를 추가하여 보자.

```
public class Cage {
   ArrayList<Tiger> list = new ArrayList<Tiger>();
   ...
}
```

2. 카드 게임 프로그램을 컬렉션 라이브러리를 이용하여서 작성하여 보자.

❶ 카드를 나타내는 Card 클래스를 작성한다. 하나의 카드는 무늬와 숫자를 가지고 있어야 한다.

```
import java.util.*;
import java.util.Collections;

class Card {
      String suit;                    // 무늬
      String number;                  // 숫자

      public Card(String suit, String number) {
         _____________ = suit;
         _____________ = number;
      }
      public String toString(){
         return _____________________;
      }
}
```

❷ 카드 데크를 나타내는 CardDeck 클래스를 작성한다. CardDeck은 총 52개의 Card 객체를 저장하고 있어야 한다. ArrayList를 사용하여서 카드들을 저장하도록 하자. 생성자에서 모든 카드를 생성한다.

```
class CardDeck {
      ArrayList<Card> deck = new _________________;

      // 52개의 카드를 생성하여 덱에 넣는다
      public CardDeck() {
         String[] suit = { "Clubs", "Diamonds", "Hearts", "Spades" };
         String[] number = { "2", "3", "4", "5", "6", "7", "8",
                  "9", "10", "Jack", "Queen", "King", "Ace" };
```

```
        for (int i = 0; i < suit.length; i++)
            for (int j = 0; j < number.length; j++)
                deck._____(new Card(suit[i], number[j]));
    }
```

❸ CardDeck 클래스에 카드를 섞는 메소드인 shuffle()을 추가한다. Collections 클래스의 정적 메소드인 shuffle()을 사용하라.

```
    public void shuffle() {
        Collections._____(deck);
    }
```

❹ 데크의 처음에서 카드를 제거하여서 반환하는 메소드인 deal()을 작성한다.

```
    public Card deal() {
        return deck._____;
    }
}
```

❺ 카드 경기자를 나타내는 Player 클래스를 구현한다. Player도 카드의 리스트를 가지고 있어야 한다.

```
class Player {
    _____;// Card의 리스트 생성
```

❻ Player는 카드를 얻어서 자신의 리스트에 추가하는 메소드인 getCard()를 가지고 있다.

```
    public void getCard(Card card){
        list._____(card);              // 자신의 리스트에 추가
    }
    public void showCards(){
        System.out.println(list);
    }
}
```

❼ 드디어 main()을 작성하여 보자. 먼저 CardDeck 객체를 생성하고 데크의 카드를 섞은 후에 경기자를 두 명 생성한다. 카드를 한장씩만 분배하고 각자가 가진 카드를 화면에 표시한다.

```
public class CardGame
{
        public static void main(String[] args) {
                CardDeck deck = new CardDeck();
                deck.shuffle();
                Player p1 = new Player();
                Player p2 = new Player();
                p1.getCard(deck.deal());
                p2.getCard(deck.deal());
                p1.showCards();
                p2.showCards();
    }
}
```

8 실제 포커 게임이 될 수 있도록 여러 가지 경기 규칙을 추가하여 보라. 즉 원페어, 투페어, 스트레이트나 플러쉬를 판별하여서 어떤 경기자가 승리하였는지를 화면에 표시하라.