선발 시험

주의 사항

- 전체 문제를 한번 보고, 가능한 문제 우선으로 작성합니다.
- 각 문제는 **부분 점수 부여** 가 될 수 있습니다.
- 제출시 소스 코드만 제출되도록 정리하고, **폴더 전체를 압축** 해 제출합니다.
  - 제출시 **폴더명은 "이름_학교"** 로 생성합니다.
  - 제출할 파일은 정상적으로 압축되었는지 **반드시 확인 후 제출** 합니다.
- 제출은 **nhnexam@gmail.com** 으로 시험 종료 시간 전에 도착하도록 전송합니다.
- 본 시험은 과정 선발보다 **응시자 평가를 우선** 으로 합니다.
- **생성형 AI를 이용할 경우**, 정확한 평가가 되지 않아 여러분께 도움이 되지 않습니다.
- 예제 코드에서
  - /* … **/ 로 표기된 부분은 *코드의 일부만 추가되는 부분** 이며,
  - //… 로 표기된 부분은 **여러 문장이 포함** 될 수 있습니다.

문제 1. arraycopy를 구현 하라. (10)

System.arraycopy와 동일한 함수를 만들고, 다음의 코드가 정상적으로 수행되는지 확인해 보자.
단, arraycopy내에서는 외부 함수 호출이나, 추가 배열 사용이 없어야 한다.

```
package example;

import java.util.Arrays;

public class Exam1 {
   static void arraycopy(int[] src, int srcPos, int[] dest, int destPos, int length) {
      //...
   }

   public static void main(String[] args) {
      int[] a = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
      int[] b = a.clone();
      int[] c = new int[a.length];
      int[] d = new int[a.length];

      // 1-1 서로 다른 배열로 복사(5)
      System.out.println("a : " + Arrays.toString(a));
      System.out.println("b : " + Arrays.toString(b));
      System.out.println("c : " + Arrays.toString(c));
      System.out.println("d : " + Arrays.toString(d));
      System.out.println("System.arraycopy(a, 1, c, 3, 5)");
      arraycopy(a, 1, c, 3, 5);
      System.out.println("c : " + Arrays.toString(c));
```

```
        System.out.println("arraycopy(b, 1, d, 3, 5)");
        System.arraycopy(b, 1, d, 3, 5);
        System.out.println("d : " + Arrays.toString(d));
        System.out.println("Arrays.equals(c, d) = " + Arrays.equals(c, d));

        // 1-2 같은 배열로 복사(5)
        System.out.println("System.arraycopy(a, 1, a, 3, 5)");
        System.arraycopy(a, 1, a, 3, 5);
        System.out.println("arraycopy(b, 1, b, 3, 5)");
        arraycopy(b, 1, b, 3, 5);
        System.out.println("Arrays.equals(a, b) = " + Arrays.equals(a, b));
        System.out.println("System.arraycopy(a, 4, a, 2, 5)");
        System.arraycopy(a, 4, a, 2, 5);
        System.out.println("arraycopy(b, 4, b, 2, 5)");
        arraycopy(b, 4, b, 2, 5);
        System.out.println("Arrays.equals(a, b) = " + Arrays.equals(a, b));
    }
}
```

결과는 아래와 같다.

```
a : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
b : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
c : [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
d : [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
System.arraycopy(a, 1, c, 3, 5)
c : [0, 0, 0, 1, 2, 3, 4, 5, 0, 0]
arraycopy(b, 1, d, 3, 5)
d : [0, 0, 0, 1, 2, 3, 4, 5, 0, 0]
Arrays.equals(c, d) = true
System.arraycopy(a, 1, a, 3, 5)
arraycopy(b, 1, b, 3, 5)
Arrays.equals(a, b) = true
System.arraycopy(a, 4, a, 2, 5)
arraycopy(b, 4, b, 2, 5)
Arrays.equals(a, b) = true
```

문제 2. OrderedList class를 구현하라. (30)

OrderedList class는 아래에 정의되어 있는 List interface를 구현한다.

```
interface List {
    public void insert(int value);
    public int getFirst();
    public int removeFirst();
    public boolean isEmpty();
```

```
}
```

OrderedLinkedList class를 다음을 참고하여 정의해 보자.

```
class OrderedLinkedList implements List {
  class Node {
    int value;
    Node next;

    //...
  }

  //...

  public void insert(int value) {
    //...
  }

  public int getFirst() {
    //...
  }

  public int removeFirst() {
    //...
  }

  public boolean isEmpty() {
    //...
  }

  public static void main(String[] args) {
    OrderedLinkedList list = new OrderedLinkedList();

    System.out.println("isEmpty : " + list.isEmpty());
    list.insert(10);
    list.insert(5);
    System.out.println("getFirst : " + list.getFirst());
    list.insert(7);
    list.removeFirst();
    System.out.println("isEmpty : " + list.isEmpty());
    System.out.println("getFirst : " + list.getFirst());
    list.insert(12);
    list.insert(3);
    System.out.println("getFirst : " + list.getFirst());
    list.removeFirst();
    System.out.println("getFirst : " + list.getFirst());
    list.removeFirst();
    System.out.println("getFirst : " + list.getFirst());
```

```
        list.removeFirst();
        System.out.println("isEmpty : " + list.isEmpty());
    }
}
```

결과는 아래와 같다.

```
isEmpty : true
getFirst : 5
isEmpty : false
getFirst : 7
getFirst : 3
getFirst : 7
getFirst : 10
getFirst : 12
isEmpty : true
```

- Node class를 Inner class로 정의하여 데이터 관리에 사용한다.
- 입력되는 값은 오른 차순으로 정렬된다.

문제 3. OrderedArrayList를 두개의 배열을 이용해 구현하라. (30)

- OrderedLinkedList에서와 달리 inner class를 사용하지 않는다.
- 초기 배열 크기를 갖는다.
    - 배열 크기는 OrderedArrayList에 넣을 수 있는 요소의 최대수와 동일하다.

```
class OrderedArrayList implements List {
    int [] values;
    int [] nexts;
    //...

    public static void main(String[] args) {
        OrderedArrayList list = new OrderedArrayList(10);

        System.out.println("isEmpty : " + list.isEmpty());
        list.insert(10);
        list.insert(5);
        System.out.println("getFirst : " + list.getFirst());
        list.insert(7);
        list.removeFirst();
        System.out.println("isEmpty : " + list.isEmpty());
        System.out.println("getFirst : " + list.getFirst());
        list.insert(12);
        list.insert(3);
```

```
        System.out.println("getFirst : " + list.getFirst());
        list.removeFirst();
        System.out.println("getFirst : " + list.getFirst());
        list.removeFirst();
        System.out.println("getFirst : " + list.getFirst());
        list.removeFirst();
        System.out.println("isEmpty : " + list.isEmpty());
    }
}
```

문제 4. OrderedArrayList에 더이상의 요소를 추가할 수 없을 경우, RuntimeException을 발생시켜라. (10)

```
class OrderedArrayList implements List {
    int [] values;
    int [] nexts;
    //...

    public static void main(String[] args) {
        OrderedArrayList list = new OrderedArrayList(5);

        list.insert(1);
        System.out.println("1을 넣었습니다.");
        list.insert(2);
        System.out.println("2를 넣었습니다.");
        list.insert(3);
        System.out.println("3을 넣었습니다.");
        list.insert(4);
        System.out.println("4를 넣었습니다.");
        list.insert(5);
        System.out.println("5를 넣었습니다.");

        try {
            list.insert(6);
        } catch(RuntimeException ignore) {
            System.out.println("공간이 부족해 6을 추가할 수 없습니다.");
        }

        System.out.println(list.removeFirst() + "을/를 제거하였습니다.");
        System.out.println(list.removeFirst() + "을/를 제거하였습니다.");
        System.out.println(list.removeFirst() + "을/를 제거하였습니다.");
        System.out.println(list.removeFirst() + "을/를 제거하였습니다.");
        System.out.println(list.removeFirst() + "을/를 제거하였습니다.");
        try {
            System.out.println(list.removeFirst() + "을/를 제거하였습니다.");
        } catch(RuntimeException ignore) {
```

```
            System.out.println("list에 제거할 요소가 없습니다.");
        }
    }
}
```

결과는 아래와 같다.

```
1을 넣었습니다.
2를 넣었습니다.
3을 넣었습니다.
4를 넣었습니다.
5를 넣었습니다.
공간이 부족해 6을 추가할 수 없습니다.
1을/를 제거하였습니다.
2을/를 제거하였습니다.
3을/를 제거하였습니다.
4을/를 제거하였습니다.
5을/를 제거하였습니다.
list에 제거할 요소가 없습니다.
```

문제 5. List를 다음과 같이 수정할 경우, OrderedLinkedList를 구현하라. (50)

- 테스트 코드 작성
  - 테스트에서 사용되는 데이터는 학생 정보 클래스 (Student class)
    - 학번, 이름
    - 학번 기준 정렬
- 재정의 함수에서 파라메터가 다를 경우 컴파일러 오류가 나지 않더라도 감점

```
interface List<T> {
    public void insert(T value);
    public T getFirst();
    public T removeFirst();
    public boolean isEmpty();
}
```

```
class OrderedLinkedList<T extends Comparable<T>> implements List<T> {
    //...


    public static void main(String[] args) {
        OrderedLinkedList<Student> students = new OrderedLinkedList<>();
```

```
        students.insert(new Student("s2", 2));
        students.insert(new Student("s3", 3));
        students.insert(new Student("s1", 1));

        System.out.println(students.removeFirst().getId());
        System.out.println(students.removeFirst().getId());
        System.out.println(students.removeFirst().getId());

    }
}
```

결과는 아래와 같다.

```
1
2
3
```

문제 6. 다음 코드를 완성하라. (70)

Shape

- 평면 또는 공간상에서 표현되는 모든 도형을 포함한다.
- 문자열 출력시 도형의 종류를 출력한다. (클래스 이름)
- Shape 객체는 생성되지 않는다.

Shape2D

- 2차원 평면 도형을 나타낸다.
- 면적과 그리기 기능을 지원해야 한다.

```
interface Shape2D {
    double getArea();

    void draw();
}
```

Shape3D

- 3차원 평면 도형을 나타낸다.
- 부파과 만들기 기능을 지원해야 한다.

```
interface Shape3D {
    double getVolume();

    void build();
```

```
}
```

Circle

- 평면상에 그려지는 원을 나타낸다.
- 생성시 반지름이 주어져야 한다.
- 원의 면적은 Math.PI * Math.pow(r,2)
- toString을 재정의하지 않는다.

```
class Circle /*...*/ {
    // ...
}
```

Rectangle

- 평면상에 그려지는 사각형을 나타낸다.
- 생성시 폭과 높이가 주어져야 한다.
- toString을 재정의하지 않는다.

```
class Rectangle /*...*/ {
    // ...
}
```

Ball

- 공간상에 만들어지는 구를 나타낸다.
- 생성시 구의 반지름이 주어져야 한다.
- 구의 부피는 4/3 * Math.PI * Math.pow(r,3)
- toString을 재정의하지 않는다.

```
class Ball /* ... */ {
    // ...
}
```

Box

- 공간상에 만들어지는 박스를 나타낸다.
- 생성시 폭, 높이, 깊이가 주어져야 한다.
- toString을 재정의하지 않는다.

```
class Box /* ... */ {
    // ...
}
```

```
class World {
    public static void main(String[] args) {
        Shape[] shapes = new Shape[4];

        // 6-1 Circle을 구현하라. (5)
        Circle circle = new Circle(3);
        System.out.println(circle + "의 넓이는 " + circle.getArea() + " 입니다.");

        // 6-2 Box을 구현하라. (5)
        Box box = new Box(7, 8, 9);
        System.out.println(box + "의 부피는 " + box.getVolume() + " 입니다.");

        shapes[0] = circle;
        shapes[1] = new Rectangle(4,5);
        shapes[2] = new Ball(6);
        shapes[3] = box;

        // 6-3 도형 종류를 출력하라(10)
        System.out.println(Arrays.toString(shapes));

        // 6-4 2차원 도형만 출력하라(20)
        System.out.print("2차원 도형은 ");
        for (Shape /* ... */) {
            // ...
        }
        System.out.print("이며, 넓이는 ");
        for (Shape /* ... */) {
            // ...
        }
        System.out.println("입니다.");

        // 6-5 3차원 도형만 출력하라(30)
        // 주석 처리된 부분에만 코드를 추가한다.
        // Java Stream을 이용해 구현한다.
        Object[] shape3ds = /* ... */;
        System.out.println(/* ... */);
    }
}
```

결과는 아래와 같다.

```
Circle의 넓이는 28.274333882308138 입니다.
Box의 부피는 504.0 입니다.
Shapes : [Circle, Rectangle, Ball, Box]
2차원 도형은 Circle Rectangle 이며, 넓이는 28.274333882308138 20.0 입니다.
3차원 도형은 [Ball, Box] 이며, 부피는 [904.7786842338603, 504.0] 입니다.
```