EXERCISE
INTRODUCTION TO JAVA PROGRAMMING

1. 다음 문장이 참인지 거짓인지를 말하라. 만약 거짓이면 올바르게 수정하라.

   ❶ 클래스는 오직 하나의 인터페이스만을 구현할 수 있다.

   ❷ 디폴트로 클래스 안의 모든 메소드는 public이다.

   ❸ 클래스가 추상 메소드를 하나라도 가지고 있으면 그 클래스로는 객체를 생성할 수 없다.

2. AudioSystem 클래스 정의의 첫 번째 문장만을 쓰시오. AudioSystem은 SoundSystem을 확장하고 MP3playable 인터페이스와 TurnTablePalyable 인터페이스를 구현한다.

3. 다음의 추상 클래스 정의에서 잘못된 점을 모두 지적하라.

```
abstract public class Shape  {
   int x, y;
   public abtract void draw()
   public abtract void move() {   }
}
```

4. 추상 메소드 sound()를 가지고 있는 추상 클래스 Bird를 작성하고 Bird를 상속받아서 Dove 클래스를 작성하라. Dove 클래스의 sound()에서는 "coo coo"를 출력한다.

5. 다음의 인터페이스 선언과 사용에서 잘못된 점을 모두 지적하라.

```
public interface Edible {
   boolean amount;
   final int TYPE=10;
   public abtract void eat() {   };
}
public class Sandwitch extends Edible {
   public void eat() {   }
}
```

6. 클래스 Rectangle이 Drawable 인터페이스와 Movable 인터페이스를 모두 구현하려고 한다. 다음 중 맞게 선언한 것은?

❶ public class Rectangle extends Drawable, Movable

❷ public class Rectangle implements Drawable, Movable

❸ public class Rectangle inherits Drawable, Movable

❹ public class Rectangle implements Drawable extends Movable

---

7. MyClass가 인터페이스 InterA와 InterB를 구현한다고 하자.

```
MyClass obj = new MyClass();
InterA a;
InterB b;
```

다음의 대입문 중에서 오류가 발생하거나 형변환이 필요한 것을 지적하고 바르게 수정하시오.

```
a = obj;
b = obj;
b = (InterB) a;
obj = a;
obj = b;
b = a;
```

8. 만약 클래스 Desk가 Movable 인터페이스를 구현한다고 하자. 다음의 대입문 중에서 적법한 것은?

```
❶ Desk desk = new Desk();
❷ Movable m = desk;
❸ Rectangle box = new Rectangle();
❹ m = box;
❺ m = (Movable) box;
❻ desk = m;
❼ desk = (Desk) m;
❽ desk = (Desk) box;
```

9. 다음의 내부 클래스의 메소드 m()이 접근할 수 있는 변수들을 나열하라.

```
public class MyClass {
   private int a;
   protected int b;
   int c;

   class MyInner {
      int d;
      public void m() {
         ...
      }
   }
}
```

10. 다음 문장의 참과 거짓을 말하라.

❶ 참조 변수의 타입이 아니라 객체의 타입에 따라서 호출되는 메소드가 결정된다.

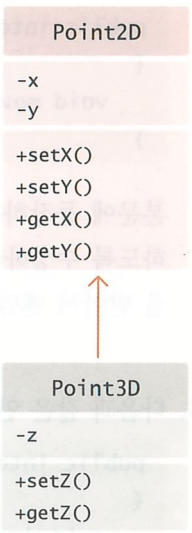
❷ 수퍼 클래스의 참조 변수는 서브 클래스의 객체를 참조할 수 있다.

❸ 서브 클래스의 참조 변수는 수퍼 클래스의 객체를 참조할 수 있다.

❹ 클래스는 하나의 인터페이스만을 구현할 수 있다.

❺ 디폴트로 인터페이스의 모든 필드와 메소드는 public이다.

11. 오른쪽 URL 그림과 같은 클래스 상속 관계를 가정하라.



❶ 다음과 같은 문장은 적법한가? 그 이유는?

```
Point2D p = new Point3D();
```

❷ 다음과 같은 문장은 적법한가? 만약 적법하지 않다면 적법하도록 고쳐보라.

```
p.setX(100);
p.setZ(40);
```

❸ 다음과 같은 문장은 적법한가? 그 이유는?

```
Point2D p = new Point3D();
```

❹ Point2D와 Point3D 클래스를 실제로 작성하여 보라.

PROGRAMMING
INTRODUCTION TO JAVA PROGRAMMING

1. 다음과 같은 인터페이스들을 정의하라.

```
public interface Buyable
{
   public int getPrice();
}
```

Television 클래스와 Refrigerator 클래스는 Buyable 인터페이스를 구현한다. Television 클래스와 Refrigerator 클래스는 모델명, 메이커, 가격을 필드로 가진다. 필요한 접근자와 설정자를 정의하라. 필드의 초기값을 매개 변수로 전달받는 생성자를 정의하라. 몇 개의 객체를 생성한 후에 다음과 같은 메소드를 정의하여서 가격을 출력하여 보자.

```
private static void printPrice(Buyable item)
{
   System.out.printf("%10d", item.getPrice());
}
```

2. 다음과 같은 인터페이스들을 정의하라.

```
public interface Movable
{
   void move();
}
```

본문에 등장하는 2차원 도형인 원, 사각형, 삼각형 등이 위의 인터페이스를 구현하도록 수정하라. move() 메소드는 도형의 기준점을 이동한다. Movable 객체 배열을 받아서 랜덤하게 객체를 이동시키는 프로그램을 작성하라.

3. 다음과 같은 인터페이스들을 정의하라.

```
public interface Drawable
{
   void draw();
}
```

본문에 등장하는 2차원 도형인 원, 사각형, 삼각형 등이 위의 인터페이스를 구현하도록 수정하라. draw() 메소드에서는 실제로 그리지는 않고 메시지만을 출력하라. Drawable 객체 배열을 받아서 랜덤하게 draw()를 호출하는 프로그램을 작성하라.

4. 다음과 같은 인터페이스를 정의하라.

```
public interface filter
{
   boolean isItOK(Object obj);
}
```

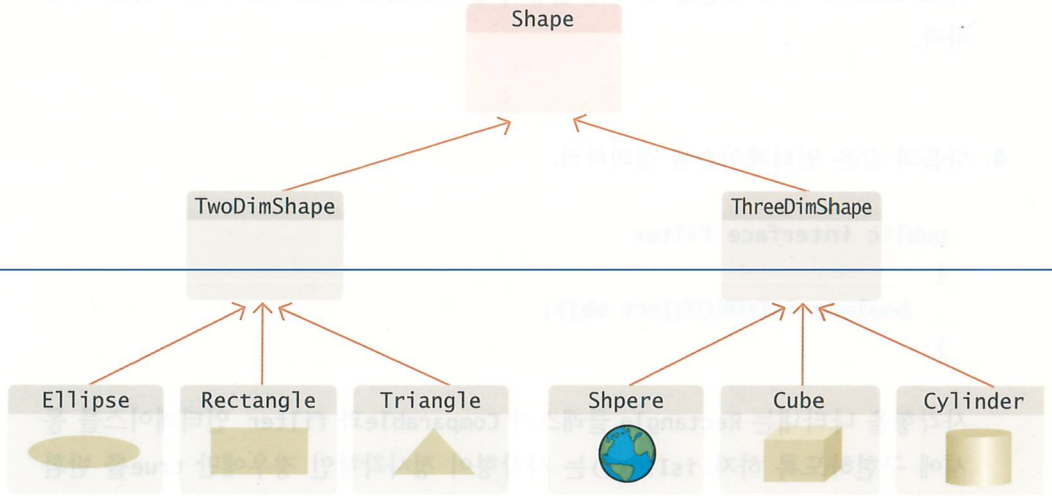
사각형을 나타내는 Rectangle 클래스가 Comparable과 filter 인터페이스를 동시에 구현하도록 하자. isItOk()는 사각형이 정사각형인 경우에만 true를 반환하도록 하라.

5. Person이라는 클래스를 정의하라. Person은 이름(name)과 키(height)를 필드로 가진다. Person은 Comparable 인터페이스를 구현한다. 이 Comparable 인터페이스를 이용하여서 가장 키 큰 사람의 이름을 반환하는 메소드 getMaximum(Person[] array)을 구현하고 테스트하라.

6. 10장 배열에 등장하였던 선택 정렬 코드를 인터페이스를 이용하여 다시 작성하여 보자. SelectionSort라고 하는 클래스를 작성하고 여기에 정적 메소드인 sort()를 추가한다. sort()을 이용하여 정렬하기 위해서는 대상 객체가 Comparable 인터페이스를 구현하여야 한다. 메소드 void static sort(Comparable[] array)을 구현하고 테스트하라.

7. 도형에 관한 클래스를 다형성을 이용하여서 작성하여 보자. Shape 클래스는 전체 도형의 수퍼 클래스이다. Shape에서 상속된 TwoDimShape은 2차원 도형을 나타내며 면적을 계산하기 위하여 getArea()라는 메소드를 가진다. Shape에서 상속된 ThreeDimShape은 3차원 도형을 나타내며 체적을 계산하기 위하여 getVolume()을 가진다. 2차원 도형에는 타원, 사각형, 삼각형이 있으며 3차원 도형에는 구, 직육면체, 원기둥이 있다. 다양한 도형을 생성하여 Shape 배열에 저장하는 프로그램을

작성하라. 각 도형은 자신의 정보를 텍스트로 출력한다. 프로그램의 반복 루프에서 각 도형이 2차원 도형이면 면적을 출력하고 3차원 도형이면 체적을 출력한다.



8. 3차원 도형들을 칠하는데 필요한 페인트의 양을 계산하는 프로그램을 작성하여 보자. 수퍼 클래스는 Shape이며 자식 클래스는 구(Sphere), 직육면체(Cube), 원기둥(Cylinder)이다. Shape은 이름(name)과 3차원 도형의 표면적을 계산하는 메소드 getSurfaceArea() 만을 가지고 있다. 하나의 구, 하나의 직육면체, 하나의 원기둥을 생성하여서 Shape 배열에 저장하고 저장된 모든 도형에 대하여 표면적을 계산하여 보자.

9. 마트에서 판매되는 물건을 클래스 Product로 나타내자. Product 클래스에서 상속받아서 할인 물건을 나타내는 DiscountProduct 클래스를 작성할 수 있다.

❶ Product 클래스는 물건의 이름(name)과 가격(price)를 필드로 가지고 있다. 또한 현재 물건의 가격을 계산하는 getPrice() 메소드를 가지고 있다. Product 클래스를 구현하고 테스트하여 보자. 생성자와 필드에 대한 설정자와 접근자를 작성하라. toString()에서 물건의 이름과 가격을 출력한다.

❷ DiscountProduct는 Product로부터 상속받는다. %단위의 할인율(discount)를 필드로 가진다. getPrice() 메소드는 할인된 가격을 반환하도록 재정의된다. DiscountProduct 클래스를 구현하고 테스트하여 보자. 생성자와 필드에 대한 설정자와 접근자를 작성하라. toString()에서 물건의 이름과 할인된 가격을 출력한다.

❸ main() 메소드 안에서 다음과 같은 문장을 사용하여서 어떤 getPrice()가 호출되는지를 관찰하라.

```
Product p1 = new Product("toothbrush", 3000);
Product p2 = new DiscountProduct("toothbrush", 3000, 15%);
System.out.println(p1.getPrice());
System.out.println(p2.getPrice());
```

10. 책을 대여해주는 업체를 위한 Book이라는 클래스를 작성한다. Book은 관리번호(number), 제목(title), 저자(author)를 필드로 가진다. Book 클래스는 각 필드에 대한 접근자와 설정자를 가진다. Object 클래스의 equals() 메소드를 재정의하여서 만약 관리번호가 동일하면 동일한 책으로 간주한다. 다음으로 Book으로부터 상속을 받는 Novel, Poet, ScienceFiction 클래스를 작성한다. 이들 클래스들은 연체된 날짜에 따라서 연체료를 계산하는 getLateFees() 메소드를 재정의한다. getLateFees() 메소드는 연체 일수를 매개 변수로 받는다. Novel의 경우에는 300원/일이고 Poet는 200원/일, ScienceFiction는 600원/일이다. main() 메소드에서 작성된 클래스들을 테스트하라.

LAB
INTRODUCTION TO JAVA PROGRAMMING

1. 탈것을 나타내는 Vehicle 클래스를 추상 클래스로 정의하여 보자. 먼저 다음과 같은 코드를 입력하자.

```
abstract class Vehicle  {
   public abstract double getKilosPerLiter();
}
```

❶ 추상 클래스에 필드를 추가할 수 있는지를 알아보기 위하여 Vehicle 클래스에 정수형 변수 speed를 추가하여 보자.

❷ 추상 클래스에 메소드를 추가할 수 있는지를 알아보기 위하여 현재 속도를 문자열로 출력하는 printSpeed() 메소드를 추가하여 보자.

❸ 위의 추상 클래스를 테스트하기 다음과 같은 테스트 클래스를 작성하여 보자.

```
public class CarTest  {
   public static void main(String args[]) {
      _____________________________________; // ❶
   }
}
```

❹ 추상 클래스로 객체를 생성할 수 있는지 알아보기 위하여 ❶ 위치에서 객체 생성 문장을 작성하고 컴파일하여 보자. 결과를 기록하자.

❺ 위의 추상 클래스를 상속받아서 Car 클래스를 정의하여 보자. 반드시 구현하여야 할 메소드는 무엇인가?

```
class Car extends Vehicle  {
   _____________________________________ // ❷
}
```

❻ VehicleTest 클래스의 main()에서 Car 객체를 생성하여 보자. Car 객체를 통하여 getKilosPerLiter(), printSpeed() 메소드를 호출하여 보자.

2. 다음과 같은 인터페이스를 구현하여 보자.

```
interface Movable  {
   public abstract void speedUp(int amount);
   public abstract void speedDown(int amount);
}
```

❶ 메소드를 정의할 때 abstract를 생략하여도 되는지 실험하여 보라.

❷ 인터페이스 안에 다음과 같이 속도를 나타내는 speed 변수를 추가할 수 있는지 실험하여 보자. 어떤 오류 메시지가 나오는가?

```
int speed;
```

❸ speed의 정의를 다음과 같이 변경하면 어떻게 되는가?

```
int speed = 100;
```

이 경우 speed는 어떻게 취급되는가? 다음과 같이 변경하여 보자.

```
final int speed = 100;
```

❹ 인터페이스에 메소드를 추가할 수 있는지를 알아보기 위하여 현재 속도를 문자열로 출력하는 printSpeed() 메소드를 추가하여 보자.

❺ 인터페이스를 구현한 Car 클래스를 정의하여 보자. 반드시 구현하여야 할 메소드는 무엇인가? 필요한 필드들과 TurnLeft(), TurnRight() 메소드를 추가하라.

```
class Car __________ Movable  {
   ____________________________________
}
```

❻ CarTest 클래스의 main()에서 Car 객체를 생성하여 보자. Car 객체를 통하여 speedUp(), speedDown() 메소드를 호출하여 보자.

```
public class CarTest {
   public static void main(String args[]) {
      ____________________________________; // 객체 생성
      ____________________________________; // 메소드 호출
      ____________________________________; // 메소드 호출
```

```
    }
}
```

❼ Movable 참조 변수로 Car 객체를 참조하여 보라. speedUp() 메소드를 참조하여 보자. TurnLeft()나 TurnRight() 메소드도 호출할 수 있는가?

```
Movable m = new Car();
m.speedUp();  // 가능한가?
m.TurnLeft(); // 가능한가?
```

❽ 여러 가지 형 변환을 테스트하여 보자.

```
Movable m = new Car();
Car c = m;                  // 가능한가?
Car c = (Car)m;             // 가능한가?
Car c = new Car();          // 가능한가?
Movable m = c;              // 가능한가?
Movable m = (Movable)c;     // 가능한가?
```

3. Vehicle 클래스를 상속받고 Movable 인터페이스를 구현하는 Car 클래스를 작성하여 보자. 필요한 필드들과 생성자, 메소드들을 모두 포함시켜라. 몇 개의 Car 객체를 생성하여서 테스트하라.

```
class Car _____ Vehicle _____ Movable  {
   ... // 필드, 생성자, 메소드 정의
}
```

4. 간단한 게임 프로그램을 다형성을 이용하여서 작성하여 보자. 게임에는 많은 캐릭터들이 있다. 예를 들어서 호빗, 타이탄, 주술사 등이 있다. 이들 클래스들이 공통의 수퍼 클래스 GameCharacter로부터 상속받았다고 가정하라. 이 클래스는 draw()라는 메소드를 가지고 있다. 각 서브 클래스는 이 draw() 메소드를 재정의한다. 이들 캐릭터들은 GameCharacter 배열에 참조값이 저장된다. 화면을 다시 그리기 위하여 프로그램은 주기적으로 모든 캐릭터의 draw()를 호출한다. 예를 들어서 호빗의 경우, draw()가 호출되면 콘솔에 "호빗을 그립니다"라고 출력한다.

```
class GameCharacter {
   public void draw() {   System.out.println("GameCharacter()의 draw()");      }
```

```
   public void getLife() { System.out.println("GameCharacter()의 getLife()"); }
}
class Hobitt extends GameCharacter {
   public void draw() {       System.out.println("호빗을 그립니다.");     }
   public void getRing() { System.out.println("GameCharacter()의 getRing()");    }
}
```

❶ 타이탄을 나타내는 클래스 Titan과 주술사를 나타내는 클래스 Sorcerer도 같은 방법으로 정의해보라.

❷ 이들 클래스들을 테스트할 수 있는 클래스 Test를 생성하여서 호빗, 타이탄, 주술사 객체를 생성하여 보자.

```
public class Test {
   public static void main(String arg[]) {
      GameCharacter g1 = new Hobitt();              // 호빗 객체 생성
   }
}
```

❸ g1.getLife()과 같이 호출하여 보자.

❹ g1.getRing()를 호출하여 보자. 오류가 발생한다면 그 이유는?

❺ ((Hobitt)g1).getRing()와 같이 호출하여 보자. 결과를 기록하라.

❻ 호빗, 타이탄, 주술사를 각각 2개씩 생성하고 이들을 모두 array라는 배열에 저장하라. 배열의 타입은 무엇으로 하면 좋은가?

❼ 반복 루프를 사용하여서 array 배열 안에 저장된 객체에 대하여 draw() 메소드를 호출하여 보자. 어떤 draw()가 호출되는가?

❽ ❼번을 별도의 메소드 drawAll()로 독립시켜 보자. 배열을 매개 변수로 전달하라.

❾ drawAll()의 매개 변수를 Object 타입의 배열로 할 수 있는가? drawAll()에서 instanceof 연산자를 사용하여서 각 객체의 타입을 분류하여 보자.

❿ 게임에 필요한 필드와 생성자, 접근자, 설정자 메소드를 추가로 구현해보자.

Power JAVA