이것만은 알고 갑시다 정답

이것만은 알고 갑시다 1장

1. a) 클래스, 메소드, 명령문 b) main
2. 자바 가상 기계 또는 Java Virtual Machine 또는 JVM
3. 5행에 중괄호 }가 하나 더 있어야 합니다.

이것만은 알고 갑시다 2장

1. 11행에서 max 변수를 사용한 것이 잘못입니다. max 변수는 if 문에 종속된 블록 안에 선언되어 있기 때문에 그 블록 밖에서는 사용할 수 없습니다.
2. K라는 문자가 출력됩니다. 이 프로그램이 "키위“를 출력하도록 만들려면 switch 문 안에 다음과 같은 부분을 추가해야 합니다.

```
case 'K' :
  System.out.println("키위 “);
  break;
```

3. ⓐ int cnt = 10
   ⓑ cnt > 0 또는 cnt >= 1
   ⓒ cnt--
4. ⓐ double num : arr
   ⓑ total += num; 또는 total = total + num;
   * num 대신 다른 변수 이름을 사용해도 됩니다.
5. 1) "Hello, " + args[0]
   2) java HelloEverybody World

이것만은 알고 갑시다 3장

1. byte 타입으로 표현할 수 있는 범위는 -128부터 127까지입니다. 그렇기 때문에 0부터 199까지 반복을 수행하는 for 문의 카운트 변수로 사용될 수 없습니다.
2. a) String b) double c) int d) double e) boolean f) char
3. '\r\n' : 작은 따옴표 안에는 하나의 문자만 쓸 수 있습니다.
   .5e100f : 표현 범위를 초과하는 float 타입 리터럴입니다.
   0xABCDEFABCDEF : 표현 범위를 초과하는 int 타입 리터럴입니다.
   0x12.5e2 : 16진 부동소수점수의 가수, 지수 구분은 e가 아니라 P나 p를 가지고 해야 합니다.
   "단가: ₩10000" : ₩는 escape sequence를 시작하는 문자(₩)이기 때문에 원화 표시로 사용될 수 없습니다.
   TRUE : 불리언 리터럴은 모두 소문자로 써야 합니다.
   1e-100f : float 타입으로 표현할 수 없을 정도로 미세한 값입니다. 이런 값은 float 타입의 리터럴로 사용할 수 없습니다.

이것만은 알고 갑시다 4장

1. 5행의 total = total + cnt;라는 명령문이 잘못되었습니다. total 변수는 byte 타입이고 cnt 변수는 int 타입이기 때문에 + 연산의 결과는 int 타입이 됩니다. int 타입의 값을 byte 타입 변수에 다시 대입하려고 했기 때문에 잘못입니다.
2. a) 3 b) 100 c) 10*20=200 d) true e) true f) true g) true h) 0 i) 0 j) 8
3. ++num++ : ++ 연산자가 산출하는 값은 변수가 아니기 때문에 그 결과에 다시 ++ 연산자를 사용할 수 없습니다.
   true + 3 : + 연산자는 boolean 타입과 int 타입의 피연산자에는 사용할 수 없습니다.
   true > false : > 연산자의 두 피연산자는 모두 수치 타입이어야 합니다.
   0xFF00 || 0x00FF : || 연산자의 피연산자는 모두 boolean 타입이어야 합니다.
   --100 : -- 연산자의 피연산자는 반드시 변수여야 합니다.
   120e3 << 2 : << 연산자의 피연산자는 정수 타입이어야 하는데 120e3은 부동소수점 타입 리터럴입니다.

이것만은 알고 갑시다 5장

1. 이 장에서 배웠던 방법을 사용하여 다음과 같이 상품 정보 클래스를 선언할 수 있습니다.

```
class GoodsInfo {
    String code;                // 상품코드
    String name;                // 상품명
    String maker;               // 제조사
    int price;                  // 표준단가
    double discountRate;        // 할인율
    GoodsInfo(String code, String name, String maker, int price) {
        this.code = code;
        this.name = name;
        this.maker = maker;
        this.price = price;
    }
    GoodsInfo(String code, String name, String maker, int price, double discountRate) {
        this(code, name, maker, price);
        this.discountRate = discountRate;
    }
    void updateDiscountRate(double discountRate) {     // 할인율을 변경한다
        this.discountRate = discountRate;
    }
    int getSellingPrice() {                            // 판매가를 계산한다
        return price - (int) (price * discountRate);
    }
}
```

2. final 필드의 값은 반드시 객체가 생성되는 도중에 초기화되어야 하는데 초기화되지 않았습니다. 그리고 객체가 생성된 후에는 final 필드의 값을 바꿀 수 없는데 객체를 생성하고 난 후라야 호출될 수 있는 메소드 안에서 final 필드에 값을 대입했으므로 그것도 잘못입니다.
3. width와 height 필드가 0 이하의 값을 갖게 되는 경우는 둘입니다. 첫째는 객체를 생성할 때 파라미터 값으로 0 이하의 값이 들어오는 경우고, 둘째는 객체가 생성된 후에 필드에 다른 값을 대입하는 경우입니다.
   첫 번째 경우는 객체를 생성할 때 0 이하의 값이 들어오면 익셉션이 발생되도록 하는 것으로 문제를 해결할 수 있습니다. 두 번째 경우는 객체 외부에서 필드에 다른 값을 대입하지 못하도록 필드 선언문에 private 키워드를 붙여서 문제를 해결할 수 있습니다.

```
class Rectangle {
    private int width, height;
    Rectangle(int width, int height) throws Exception {
        if (width <= 0 || height <= 0)
            throw new Exception("높이와 넓이는 플러스 값이어야 합니다.");
        this.width = width;
        this.height = height;
    }
    int getArea() {
        return width * height;
    }
}
```

그런데 이렇게 클래스를 고치면 외부에서 객체가 담고 있는 width와 height 필드 값을 읽을 수조차 없게 되므로, 이를 보완하기 위해 다음과 같이 필드 값을 리턴하는 메소드들을 추가해 주는 것이 좋습니다.

```
class Rectangle {
    private int width, height;
    Rectangle(int width, int height) throws Exception {
        if (width <= 0 || height <= 0)
            throw new Exception("높이와 넓이는 플러스 값이어야 합니다. ");
        this.width = width;
        this.height = height;
    }
    int getArea() {
        return width * height;
    }
    int getWidth() {
        return width;
    }
    int getHeight() {
        return height;
    }
}
```

4. 모든 객체가 공통적으로 접근할 데이터는 정적 필드로 선언해야 합니다. 그러므로 마지막 일련번호를 저장하는 정적 필드를 선언하고, 생성자에서 그 필드를 사용하여 seqNo 필드의 값을 설정한 후 정적 필드의 값을 1만큼 증가시키면 됩니다.

```
class BBSItem {
    static int lastSeqNo = 0;      // 마지막 일련번호
    int seqNo;                     // 일련번호
    String writer;                 // 작성자
    String writtenDate;            // 작성일자
    String title;                  // 제목
    String content;                // 내용
    BBSItem(String writer, String writtenDate, String title, String content) {
        this.seqNo = ++lastSeqNo;
        this.writer = writer;
        this.writtenDate = writtenDate;
        this.title = title;
        this.content = content;
    }
}
```

또는 마지막 일련번호 대신 다음 일련번호를 저장하는 정적 필드를 선언해도 됩니다.

5. 5행의 printCharacter 메소드 호출문이 잘못입니다. 정적 메소드 안에서는 인스턴스 메소드를 호출할 수 없기 때문입니다.

이것만은 알고 갑시다 6장

1. 주어진 음악 CD 클래스는 다음과 같이 선언할 수 있습니다.

```
class MusicCDInfo extends CDInfo {
    String artist;          // 아티스트
    String songName[];      // 곡명
    MusicCDInfo(String registreNo, String title, String artist, String[] songName) {
        super(registreNo, title);
        this.artist = artist;
        this.songName = songName;
    }
}
```

2. a) extends Rectangle
   b) super(sideLength, sideLength)
   c) getWidth() 또는 getHeight()

3. 13행과 14행에 있는 obj.borrower와 obj.checkOutDate 부분이 잘못입니다. 객체가 인터페이스 변수에 대입되어 있을 때는 그 인터페이스에 속하는 구성요소만 사용할 수 있기 때문입니다. borrower와 checkOutDate 필드는 Lendable 인터페이스의 구성요소가 아니므로 잘못입니다. 오류를 고치려면 이 부분을 파라미터 변수 이름인 borrower와 date로 바꾸어야 합니다.
4. title, senderName 필드는 final static 키워드가 없지만 인터페이스 안에 있기 때문에 컴파일할 때 자동으로 final static 키워드가 붙어서 상수 필드가 됩니다. 그런데 인터페이스의 상수 필드는 선언을 할 때 반드시 초기값을 대입해야 하는데 그렇게 하지 않은 것이 첫 번째 잘못입니다. 두 번째 잘못은 생성자입니다. 인터페이스는 생성자를 가질 수 없기 때문입니다.

이것만은 알고 갑시다 7장

1. 표시된 부분에 다음과 같은 if 문을 넣으면 됩니다.

```
if (obj instanceof CheckingAccount) {
    CheckingAccount obj2 = (CheckingAccount) obj;
    System.out.println("카드번호:" + obj2.cardNo);
}
```

2. 다음과 같이 선언할 수 있습니다.

```
enum Color {
    YELLOW, RED, BLUE
}
```

   Color, YELLOW, RED, BLUE라는 식별자 대신 다른 식별자를 사용할 수도 있습니다.

3. 다음과 같이 수정할 수 있습니다.
   [대출 가능 인터페이스]

```
interface Lendable {
    enum BookState {
        STATE_BORROWED, STATE_NORMAL
    }
    void checkOut(String borrower, String date);
    void checkIn();
}
```

[단행본 클래스]

```
class SeparateVolume implements Lendable {
    String requestNo;        // 청구번호 필드
    String bookTitle;        // 제목 필드
    String writer;           // 저자 필드
    String borrower;         // 대출인 필드
    String checkOutDate;     // 대출일 필드
    BookState state;         // 대출상태 필드
    SeparateVolume(String requestNo, String bookTitle, String writer) {
        this.requestNo = requestNo;
        this.bookTitle = bookTitle;
        this.writer = writer;
        this.state = BookState.STATE_NORMAL;
    }
    public void checkOut(String borrower, String date) {      // 대출한다
        if (state != BookState.STATE_NORMAL)
            return;
        this.borrower = borrower;
        this.checkOutDate = date;
        this.state = BookState.STATE_BORROWED;
        System.out.println("*" + bookTitle + " 이(가) 대출되었습니다.");
        System.out.println("대출인:" + borrower);
        System.out.println("대출일:" + date + "Wn");
    }
    public void checkIn() {                                   // 반납한다
        this.borrower = null;
        this.checkOutDate = null;
        this.state = BookState.STATE_NORMAL;
        System.out.println("*" + bookTitle + " 이(가) 반납되었습니다.Wn");
    }
}
```

열거 타입은 프로그램 내부에서 클래스로 취급되기 때문에 BookState 타입인 state 필드의 디폴트 값은 null입니다. 그렇기 때문에 생성자에서 필드 값을 BookState.STATE_NORMAL로 초기화했습니다.

이것만은 알고 갑시다 8장

1. 다음과 같이 고치면 됩니다.

```
import java.util.GregorianCalendar;
import java.util.Calendar;
class DateTime {
    public static void main(String args[]) {
        GregorianCalendar obj = new GregorianCalendar();
        int year = obj.get(Calendar.YEAR);
        int month = obj.get(Calendar.MONTH) + 1;
        int day = obj.get(Calendar.DAY_OF_MONTH);
        System.out.printf("오늘은 %d년 %d월 %d일입니다.%n", year, month, day);
    }
}
```

또는 다음과 같이 고칠 수도 있습니다.

```
import java.util.*;
class DateTime {
    public static void main(String args[]) {
        GregorianCalendar obj = new GregorianCalendar();
        int year = obj.get(Calendar.YEAR);
        int month = obj.get(Calendar.MONTH) + 1;
        int day = obj.get(Calendar.DAY_OF_MONTH);
        System.out.printf("오늘은 %d년 %d월 %d일입니다.%n", year, month, day);
    }
}
```

2. 같은 패키지 내의 서브클래스 안에서는 public, protected 구성요소뿐만 아니라 접근 제어 수식어가 없는 구성요소도 사용할 수 있습니다. 그러므로 SmartGoodsStock 클래스 안에서는 goodsCode, stockNum 필드, 생성자, addStock, subtractStock, getTockNum 메소드를 모두 사용할 수 있습니다.
3. Movable 인터페이스를 컴파일한 디렉토리에서 javap 명령을 실행하면 다음과 같은 내용을 확인할 수 있습니다.

이것만은 알고 갑시다 9장

1. 문자열 연결 연산자 +의 양쪽에 모두 문자열 리터럴이 있으면 자바 컴파일러는 길게 연결된 하나의 문자열 리터럴로 취급하기 때문에 왼쪽 프로그램은 하나의 String 객체만 생성합니다. 반면 오른쪽 프로그램은 3개의 서로 다른 문자열 리터럴 각각에 대해 String 객체가 생성될 뿐만 아니라 += 연산을 한 번 할 때마다 새로운 String 객체가 생성됩니다. 객체를 자꾸 생성하면 메모리 사용도 많아지고, 프로그램의 실행 속도도 떨어지기 때문에 왼쪽 프로그램이 오른쪽 프로그램보다 더 효율적이라고 할 수 있습니다.
2. 다음과 같이 완성할 수 있습니다.

```
class LongLongString {
    public static void main(String args[]) {
        StringBuilder sb = new StringBuilder();
        for (String str: args)
            sb.append(str);
        System.out.println(sb);
    }
}
```

3. 다음과 같이 완성할 수 있습니다.

```
import java.util.*;
class After100Days {
    public static void main(String args[]) {
        GregorianCalendar calendar = new GregorianCalendar();
        calendar.add(Calendar.DATE, 100);
        int year  = calendar.get(Calendar.YEAR);
        int month = calendar.get(Calendar.MONTH) + 1;
        int date  = calendar.get(Calendar.DATE);
        System.out.printf(year + "년" + month + "월" + date + "일");
    }
}
```

4. random.nextInt(5)

이것만은 알고 갑시다 10장

1. 빈칸을 다음과 같이 채우면 됩니다.

```
writer.println(str1);
writer.println(str2);
```

2. 빈칸을 다음과 같이 채우면 됩니다.

```
int num1 = in.readInt();
int num2 = in.readInt();
double num3 = in.readDouble();
System.out.println(num1);
System.out.println(num2);
System.out.println(num3);
```

readInt, readDouble 메소드의 호출 순서와 사용 방법만 맞으면 다른 부분은 다르게 작성해도 상관없습니다.

3. System.out.printf 메소드 호출문을 다음과 같이 완성하면 됩니다.

```
System.out.printf("계좌번호:%s%n", obj.accountNo);
System.out.printf("예금주 이름:%s%n", obj.ownerName);
System.out.printf("잔액:%d%n", obj.balance);
```

이것만은 알고 갑시다 11장

1. 빈칸을 다음과 같이 채우면 됩니다.

```
return "계좌번호: " + accountNo +
       "₩n예금주이름: " + ownerName +
       "₩n잔액: " + balance + "원";
```

2. 다음과 같은 결과가 출력됩니다.

```
obj1 = (10,10)
obj2 = (50,100)
```

이것만은 알고 갑시다 12장

1. 다음과 같이 완성할 수 있습니다.

```
class MaxValue {
    public static void main(String args[]) {
        if (args.length != 2) {
            System.out.println("Usage: java MaxValue <정수1> <정수2>");
            return;
        }
        try {
            int num1 = Integer.parseInt(args[0]);
            int num2 = Integer.parseInt(args[1]);
            if (num1 > num2)
                System.out.println(num1);
            else
                System.out.println(num2);
        }
        catch (NumberFormatException e) {
             System.out.println("Usage: java MaxValue <정수1> <정수2>");
        }
    }
}
```

   int 타입 대신 long 타입을 사용하고, Integer 클래스 대신 Long 클래스를 사용하여 더 넓은 범위의 정수를 비교하는 프로그램으로 만들 수도 있습니다.

2. 올바른 프로그램입니다. 프리미티브 타입인 3.14와 true는 Object 배열에 넣는 과정에서 Double 타입과 Boolean 타입으로 자동 박싱됩니다.

이것만은 알고 갑시다 13장

1. 다음과 같이 완성하면 됩니다.

```
// 게시판 클래스
import java.util.LinkedList;
class BBS {
    LinkedList<BBSItem> items;
    BBS() {                                 // 생성자
        items = new LinkedList<BBSItem>();
    }
    void add(BBSItem item) {                // 게시글을 추가한다
        items.add(item);
    }
    void modify(int index, BBSItem item) {  // 게시글을 수정한다
        items.set(index, item);
    }
    void delete(int index) {                // 게시글을 삭제한다
        items.remove(index);
    }
}
```

2. 올바른 프로그램입니다. 5행 ~ 8행에서 addLast 메소드에 double 타입의 파라미터를 넘겨주었지만 이 값은 자동으로 Double 타입으로 변환됩니다(12-3 절 자동 Boxing 참조). 그리고 10행에서 removeLast 메소드가 리턴하는 Double 값을 double 타입의 변수에 대입했지만 이 값 역시 double 타입으로 자동 변환되기 때문에(12-3 절 자동 Unboxing 참조) 문제가 되지 않습니다.
3. hashCode, equals

이것만은 알고 갑시다 14장

1. a) getProperties b) getProperty c) arraycopy d) exit e) err
2. 다음과 같이 빈칸을 채워 넣으면 됩니다.

```
System.arraycopy(arr1, 2, arr2, 0, 5);
```

3. 다음과 같이 완성하면 됩니다.

```
class OSInfo {
    public static void main(String args[]) {
        String osName = System.getProperty("os.name");
        String osVersion = System.getProperty("os.version");
        System.out.printf("%s version %s %n", osName, osVersion);
    }
}
```

이것만은 알고 갑시다 15장

1. a) unchecked exception
   b) 에러
   c) checked exception
   d) unchecked exception
   e) checked exception
2. poem.txt 파일이 없으면 FileReader 생성자에서 FileNotFoundException이 발생하고, 그 익셉션에 의해 프로그램의 실행 흐름은 15행에 있는 catch 블록으로 이동합니다. 이 catch 블록에서 "파일이 존재하지 않습니다."라는 메시지를 출력한 다음에 프로그램의 실행 흐름은 21행에 있는 finally 블록으로 이동합니다.
   finally 불록 안에는 다시 try 문이 있고, 그 try 문 안에서 reader 변수에 대해 close 메소드를 호출하는 명령문이 있습니다. 그런데 reader 변수의 값은 null이기 때문에 여기서 NullPointerException이 발생합니다. 그 익셉션으로 인해 프로그램의 실행 흐름은 25행에 있는 catch 절로 이동합니다. 이 catch 절에서는 아무 일도 하지 않기 때문에 안쪽 try 문과 바깥쪽 try 문이 모두 완료되고 프로그램은 종료합니다.
3. 다음과 같이 완성하면 됩니다.

```
class StockShortageException extends Exception {
    StockShortageException(String str) {
        super(str);
    }
}
```

이것만은 알고 갑시다 16장

1. 다음과 같이 완성하면 됩니다.

```
import java.util.ArrayList;
class SalesReport {
    int year;         // 연도
    byte quarter;     // 4분기: 1, 2, 3, 4
    ArrayList<Record> list = new ArrayList<Record>();
    SalesReport(int year, byte quarter) {
        this.year = year;
        this.quarter = quarter;
    }
    void addRecord(String name, int num, int amount) {
        list.add(new Record(name, num, amount));
    }
    int getTotal() {
        int total = 0;
        for (Record record: list)
            total += record.num;
        return total;
    }
    class Record {
        String name;
        int num;
        int amount;
        Record(String name, int num, int amount) {
            this.name = name;
            this.num = num;
            this.amount = amount;
        }
    }
}
```

2. 다음과 같이 완성하면 됩니다.

```
class ChickenFarm {
    public static void main(String args[]) {
        class Cock extends Animal {
            void say() {
                System.out.println("꼬끼요");
            }
        }
        Cock cock = new Cock();
        cock.say();
    }
}
```

이것만은 알고 갑시다 17장

1. 직렬화 프로그램의 빈 칸은 다음과 같이 채우면 됩니다.

```
out.writeObject(obj);
```

역직렬화 프로그램의 빈 칸은 다음과 같이 채우면 됩니다.

```
(Account) in.readObject();
```

2. 다음과 같이 java.io.Serializable 인터페이스를 구현하도록 만들고, 직렬화 대상에서 제외시킬 필드에 transient 키워드를 붙여주면 됩니다.

```
class PhysicalInfo implements java.io.Serializable {
    String name;
    int age;
    transient char bloodType;
    float height, weight;
    transient float bust, waist, hip;
    PhysicalInfo(String name) {
        this.name = name;
    }
}
```

이것만은 알고 갑시다 18장

1. 다음과 같이 완성하면 됩니다.
[main 메소드를 포함하는 클래스]

```
class Signs {
    public static void main(String args[]) {
        Thread thread = new MinusThread();
        thread.start();
        for (int cnt = 0; cnt < 100; cnt++)
            System.out.print( '+' );
    }
}
```

[마이너스 부호를 출력하는 스레드 클래스]

```
class MinusThread extends Thread {
    public void run() {
        for (int cnt = 0; cnt < 100; cnt++)
            System.out.print( '-' );
    }
}
```

2. 파이를 계산하는 스레드 클래스의 빈칸은 다음과 같이 채우면 됩니다.

```
sharedArea.done = true;
```

그리고 파이를 출력하는 스레드 클래스의 빈칸은 다음과 같이 채우면 됩니다.

```
!sharedArea.done
```

3. 동전을 백만번 던지는 시뮬레이션을 하는 스레드 클래스는 다음과 같이 완성하면 됩니다.

```
class SimulThread extends Thread {
    SharedArea sharedArea;
    public void run() {
        int sum = 0;
        for (int cnt = 0; cnt < 1000000; cnt++) {
            java.util.Random random = new java.util.Random();
            boolean isHead = random.nextBoolean();
            if (isHead)
                sum++;
        }
        sharedArea.ratio = sum / 1000000.0;
        synchronized (sharedArea) {
            sharedArea.notify();
        }
    }
}
```

결과를 출력하는 스레드 클래스는 다음과 같이 완성하면 됩니다.

```
class ResultThread extends Thread {
    SharedArea sharedArea;
    public void run() {
        if (sharedArea.ratio == null) {
           try {
                synchronized (sharedArea) {
                   sharedArea.wait();
                }
            }
            catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
        System.out.println(sharedArea.ratio);
    }
}
```

이것만은 알고 갑시다 19장

1. 다음과 같이 완성하면 됩니다.

```
import java.awt.*;
import javax.swing.*;
import javax.swing.table.*;
class ContactInfoFinder {
    public static void main(String[] args) {
        JFrame frame = new JFrame("연락처 검색 프로그램");
        frame.setPreferredSize(new Dimension(500, 200));
        frame.setLocation(500, 400);
        Container contentPane = frame.getContentPane();
        JTextField text1 = new JTextField(6);
        JTextField text2 = new JTextField(10);
        JTextField text3 = new JTextField(5);
        JButton button = new JButton("검색");
        JPanel panel = new JPanel();
        panel.add(new JLabel("이름"));
        panel.add(text1);
        panel.add(new JLabel("주소"));
        panel.add(text2);
        panel.add(new JLabel("전화번호"));
        panel.add(text3);
        panel.add(button);
        contentPane.add(panel, BorderLayout.NORTH);
        String colNames[] = { "이름", "주소", "전화번호" };
        DefaultTableModel model = new DefaultTableModel(colNames, 0);
        JTable table = new JTable(model);
        contentPane.add(new JScrollPane(table), BorderLayout.CENTER);
        button.addActionListener(new SearchActionListener(table, text1, text2, text3));
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);
    }
}
```

2. 다음과 같이 완성하면 됩니다.

```
import java.awt.event.*;
import javax.swing.*;
import java.util.Random;
class ThrowActionListener implements ActionListener {
    JLabel label;
    ThrowActionListener(JLabel label) {   // 생성자
        this.label = label;
    }
    public void actionPerformed(ActionEvent e) {
        Random random = new Random();
        int head = 0, tail = 0;
        for (int cnt = 0; cnt < 100; cnt++) {
            boolean isHead = random.nextBoolean();
            if (isHead)
                head++;
            else
                tail++;
        }
        label.setText("앞면:" + head + " 뒷면:" + tail);
    }
}
```

3. 다음과 같이 완성하면 됩니다.

```
import java.awt.*;
import javax.swing.*;
class DrawingPanel extends JPanel {
    int num1;     // O형의 수
    int num2;     // A형의 수
    int num3;     // B형의 수
    int num4;     // AB형의 수
    public void paint(Graphics g)  {
        g.clearRect(0, 0, getWidth(), getHeight());
        if ((num1 < 0) || (num2 < 0) || (num3 < 0) || (num4 < 0))
            return;
        int total = num1 + num2 + num3 + num4;
        if (total == 0)
            return;
        int arc1 = (int) 360.0 * num1 / total;
        int arc2 = (int) 360.0 * num2 / total;
        int arc3 = (int) 360.0 * num3 / total;
```

```
        g.setColor(Color.YELLOW);
        g.fillArc(50, 20, 200, 200, 0, arc1);
        g.setColor(Color.RED);
        g.fillArc(50, 20, 200, 200, arc1, arc2);
        g.setColor(Color.BLUE);
        g.fillArc(50, 20, 200, 200, arc1 + arc2, arc3);
        g.setColor(Color.GREEN);
        g.fillArc(50, 20, 200, 200, arc1 + arc2 + arc3, 360-(arc1 + arc2 + arc3));
        g.setColor(Color.BLACK);
        g.setFont(new Font("굴림체", Font.PLAIN, 12));
        g.drawString(" O형: 노랑", 300, 150);
        g.drawString(" A형: 빨강", 300, 170);
        g.drawString(" B형: 파랑", 300, 190);
        g.drawString("AB형: 초록", 300, 210);
    }
    void setNumbers(int num1, int num2, int num3, int num4) {
        this.num1 = num1;
        this.num2 = num2;
        this.num3 = num3;
        this.num4 = num4;
    }
}
```

이것만은 알고 갑시다 20장

1. 빈칸을 다음과 같은 명령문으로 채워 넣으면 됩니다.

```
int data = in.read();
if (data < 0)
    break;
out.write(data);
```

2. 빈칸을 다음과 같은 명령문으로 채워 넣으면 됩니다.

```
int num1 = in.readInt();
int num2 = in.readInt();
out.writeInt(num1 + num2);
```

이것만은 알고 갑시다 21장

1. 다음과 같은 create 문을 가지고 테이블을 만들고, 그 다음에 있는 insert 문으로 데이터를 입력하면 됩니다.

```
create table custinfo (
    name      varchar(10) not null,
    address   varchar(80) not null,
    phoneno   varchar(15) not null
);
```

```
insert into custinfo (name, address, phoneno) values(
     '김재영', '서울시 서초구 FFF동 222-777호', '02-540-0000');
insert into custinfo (name, address, phoneno) values(
    '박철규', '경기도 고양시 일산서구 HHH동 999-888호', '031-915-0000');
insert into custinfo (name, address, phoneno) values(
    '변재희', '서울시 마포구 EEE동 555-333호', '02-715-0000');
insert into custinfo (name, address, phoneno) values(
    '김미경', '서울시 서대문구 BCD동 888-999호', '02-326-0000');
insert into custinfo (name, address, phoneno) values(
    '진석영', '서울시 노원구 III동 777-555호', '02-977-0000');
insert into custinfo (name, address, phoneno) values(
    '박지영', '경기도 성남시 분당구 BBB동 333-444 I아파트 711동 707', '031-702-0000');
insert into custinfo (name, address, phoneno) values(
    '최미화', '인천시 계양구 DDD동 000-000호', '032-541-0000');
insert into custinfo (name, address, phoneno) values(
    '김철수', '서울시 동대문구 AAA동 11-222호', '02-958-0000');
```

2. 빈칸을 다음과 같은 명령문으로 채우면 됩니다.

```
stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("select name, address, phoneno from custinfo where "
                                 + "name like '" + toLatin1(name) + "%' and "
                                 + "address like '" + toLatin1(address) + "%' and "
                                 + "phoneno like '" + toLatin1(phoneno) + "%';");
while (rs.next()) {
    String arr[] = new String[3];
    arr[0] = toUnicode(rs.getString("name"));
    arr[1] = toUnicode(rs.getString("address"));
    arr[2] = toUnicode(rs.getString("phoneno"));
    model.addRow(arr);
}
```

3. 빈칸을 다음과 같은 명령문으로 채우면 됩니다.

```
stmt = conn.createStatement();
int rowNum = stmt.executeUpdate(
      "insert into custinfo (name, address, phoneno) values('" +
      toLatin1(name) + "', '" + toLatin1(address) + "', '" + toLatin1(phoneno) + "');");
System.out.println(rowNum + "행이 입력되었습니다.");
```

이것만은 알고 갑시다 22장

1. main 메소드를 포함하는 클래스는 다음과 같은 애플릿 클래스로 대체하면 되고, action listener 클래스는 그대로 사용해도 됩니다.

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class HelloApplet extends JApplet {
    public void init() {
        Container contentPane = getContentPane();
        JTextField text = new JTextField();
        JButton button = new JButton("확인");
        JLabel label = new JLabel("Hello");
        contentPane.add(text, BorderLayout.CENTER);
        contentPane.add(button, BorderLayout.EAST);
        contentPane.add(label, BorderLayout.SOUTH);
        ActionListener listener = new ConfirmButtonActionListener(text, label);
        button.addActionListener(listener);
    }
}
```

2. 빈칸을 다음과 같이 채우면 됩니다.

```
<HTML>
<HEAD><TITLE>헬로 프로그램</TITLE></HEAD>
<BODY>
    <APPLET CODE="HelloApplet.class" WIDTH=200 HEIGHT=50>
    </APPLET>
</BODY>
</HTML>
```