연습문제 정답

1장 연습문제

이론문제

1. 자바 소스 파일의 확장자는 .java이고 컴파일된 파일의 확장자는 .class이다.

2. ① 제임스 고슬링

3. C → C++ → Java → C#

4. 자바 언어를 처음 개발한 목적은 가전제품에 탑재되는 소프트웨어를 작성하기에 적합한 언어를 개발하는데 있었다. 가전제품은 메모리가 작고 매우 다양한 플랫폼(하드웨어 구조, 운영체제)을 가지기 때문에, 기존(C/C++ 등)의 언어로 개발하면 소프트웨어를 플랫폼마다 따로 개발하거나, 개발된 소프트웨어를 플랫폼마다 다시 컴파일하여 배포해야만 하는 어려움이 있었다. 한 번 작성하고 컴파일하여 만든 코드를 플랫폼에 관계없이 바로 실행시킬 수 있는 소프트웨어를 작성하는데 적절한 언어를 생각한 것이 바로 자바였다.

5. ① 자바 가상 기계

6. WORA(Write Once Run Anywhere)

7. ④. 자바에서는 하나의 클래스 파일(.class)에는 반드시 하나의 컴파일된 클래스만이 저장되어, 클래스가 들어 있는 클래스 파일을 찾기 쉽게 하였다.

8. 자바 응용프로그램을 개발하고자 한다면 JDK가 필요하다. JDK에는 개발자에게 필요한 각종 명령어와 JRE를 포함하고 있다.

9. ③. 자바에서는 변수나 함수는 반드시 클래스 내에 작성하여야 한다.

10. Shape.java이다. 자바에서는 클래스의 이름과 소스 파일의 이름이 일치해야 한다.

11. ⑴ A.java이다. 자바 소스 파일에 여러 개의 클래스가 작성되어 있을 때, public 속성을 가진 클래스의 이름으로 저장된다.

⑵ A.class, A$B.class, C.class, C$D.class의 4개의 클래스 파일이 생성되고, 한 클래스 파일에는 반드시 하나의 클래스만 컴파일되어 저장된다.

12. ②. 다른 3개는 플랫폼의 독립성과 관련된 설명이다.

2장 연습문제

이론문제

1. 자바에서는 클래스를 선언할 때 class 키워드를 사용한다.

2. (1) 이 프로그램은 main() 메소드가 없다. main() 메소드를 삽입하여 수정하면 다음과 같다.

```
public class SampleProgram {

    public static void main(String[] args) {
        int i;
        int j;
        i = 10;
        j = 20;
        System.out.println(i+j);
    }

}
```

(2) 클래스의 이름이 SampleProgram이므로 SampleProgram.java로 저장한다.

(3) javac SampleProgram.java

(4) java SampleProgram

3. 잘못된 식별자와 이유는 다음과 같다.

```
int %j; // %는 특수문자로 사용할 수 없다.
double 1var; // 첫 번째 문자로 숫자를 사용할 수 없다.
```

4. (1) int height;

(2) double size=0.25;

(3) double total = (double)height + size;

(4) char c = 'a';

(5) String name = "황기태";

5. (1) 67 + 12.8의 결과 값과 타입은 double 타입의 79.8
(2) 10/3의 결과 값과 타입은 int 타입의 3
(3) 10.0/3의 결과 값과 타입은 double 타입의 3.3333333333333333
(4) 10==9의 결과 값과 타입은 boolean 타입의 false

6. final double bodyTemp = 36.5;

7. 다음 각 항목의 잘못된 부분을 수정하면 다음과 같다.
(1) while(1) { } → while(true) { }
(2) int n = 3.5; → int n = (int)3.5; 혹은 double n = 3.5;
(3) int b = (3<5)?true:false; → boolean b = (3<5)?true:false;
(4) int score = 85;
if(80 < score < 90) System.out.print(score);
→ 아래와 같이 수정
int score = 85;
if(80 < score && score < 90) System.out.print(score);

8. 각 문장을 조건식으로 나타내면 다음과 같다.
(1) age는 12보다 작거나 같다. → (age <= 12)
(2) age는 b보다 작고 c보다 크다. → (age < b && age > c)
(3) age 빼기 5는 10과 같다. → (age-5 == 10)
(4) age는 b와 같지 않거나 c와 같다. → (age != b || age == c)

9. sum = (sum>100)?100:0;

10. System.out.println((n>5)?n:5);

11. text는 다음과 같고,

```
"\"를 출력하려면 \\ 다음에 \"를 붙여 \\\"과 같이 하면 됩니다."
```

화면에는 다음과 같이 출력한다.

```
System.out.println(text);
```

12. System.out.print("나는 \"Java를 " + 100 + "%\"" + "사랑해");를 실행하면 다음과 같이 출력된다.

```
나는 "Java를 100%"사랑해
```

13. (1) grade가 'A'일 때, 100, 50, 30, 10이 모두 더해져서 190이 출력된다.
(2) grade가 'B'일 때, 50, 30, 10이 모두 더해져서 90이 출력된다.
(3) grade가 'C'일 때, 30, 10이 모두 더해져서 40이 출력된다.
(4) grade가 'F'일 때, 어떤 case 문으로도 분기되지 않아 0이 출력된다.

14. switch 문으로 바꾸면 다음과 같다.

```
switch(in) {
    case "가위" : System.out.println(1); break;
    case "바위" : System.out.println(2); break;
    case "보" : System.out.println(3); break;
    default : System.out.println(0);
}
```

3장 연습문제

이론문제

1. 1에서 10 사이의 홀수를 출력하는 코드로 다음과 같이 출력된다.

```
1 3 5 7 9
```

2. ③. ③을 제외한 나머지는 모두 1에서 9까지의 합을 구하는 코드이다.

3. continue

4. i>50

5. ④ int n[3] = new int [3];. int n[] = new int [3]으로 고쳐야 한다.

6. ④ array[array.length] = 100;. 배열 array의 인덱스는 0~(배열의 크기-1)만 가능한데 array[array.length]을 사용하면 java.lang.ArrayIndexOutOfBoundsException 예외가 발생한다.

7. ⑴ char c [] = new char [10];
   ⑵ int [] n = {0,1,2,3,4,5};
   ⑶ char [] day = {'일', '월', '화', '수', '목', '금', '토' };

8. ⑴ boolean bool [] = { true, false, false, true };
   ⑵ double d [][] = new double [5][4];
   ⑶ int val [][] = {{1,2,3}, {4,5,6}, {7,8,9}, {10,11,12}};

9. ⑴ 2, 3번 라인에 컴파일 오류가 발생한다. 첫 번째 라인에서 배열에 대한 레퍼런스 myArray만 선언한 상태로, myArray는 초기화되지 않는 상태이다. 그런 상태에서 myArrayp[0], myArrayp[1]을 사용하면 없는 배열을 참조하게 되므로 컴파일 오류가 발생한다.
   ⑵ 첫 번째 라인의 코드를 int myArray[] = new int [5]; 등과 같이 배열을 할당하면 컴파일 오류가 없어진다.

10. (1)

```
char [] alpha = { 'a', 'b', 'c', 'd' };
for(int i=0; i<alpha.length; i++)
    System.out.print(alpha[i]);
```

(2)

```
char [] alpha = { 'a', 'b', 'c', 'd' };
for(char c : alpha)
    System.out.println(c);
```

11. 배열 n은 5행으로 이루어진 2차원 배열이며, 각 행에는 순서대로 1개, 3개, 1개, 4개, 2개의 원소들이 들어 있다. 이 프로그램은 각 행에 있는 원소의 개수를 다음과 같이 출력한다.

```
1 3 1 4 2
```

12. 2차원 배열 val은 다음과 같이 선언하고 초기화한다.

```
double val[][] = {{1.1, 1.2, 1.3, 1.4},
                  {2.1, 2.2},
                  {3.1, 3.2},
                  {4.1}};
```

13. ①. main()의 원형에서 abstract가 아니라 static이다.

14.

자바 프로그램이 문법에 맞지 않게 작성되었을 때, 컴파일러는 컴파일 오류를 발생시킨다. 예외는 컴파일 오류가 없는 자바 프로그램이 실행 중에 발생한 오류를 말하며, 프로그램이 이 오류에 대한 대처가 없다면 바로 종료된다. 이런 오류가 발생할 가능성이 있는 코드들을 try 블록에 넣고, 오류가 발생하면 처리할 코드는 catch 블록에 넣는다. finally 블록은 생략 가능하다.

4장 연습문제

이론문제

1. ④. 필드는 클래스의 중요한 속성을 나타내는 것으로 보호하기 위해 private으로 선언하는 것이 바람직하다.

2. ④. 생성자에서 어떤 위치에서든 return 문을 사용할 수 있다. 다만 return이 값을 리턴해서는 안 된다.

3. ③

4. ②. Book [] book = new Book [10]; 실행 결과 Book 객체에 대한 레퍼런스가 10개 만들어진다.

5. ①. void f(int a) { x = a; }와 int f(int b) { return x+b; }는 메소드 이름과 매개변수 개수 및 타입이 모두 같으므로 메소드 오버로딩이 실패한 사례이다. 리턴 타입이 다른 것은 오버로딩과 관계없다.

6. ⑴ class TV를 this()를 이용하여 수정하면 다음과 같다.

```
class TV {
    int size;
    String manufacturer;
    public TV() {
        this(32, "LG");
    }
    public TV(String manufacturer) {
        this(32, manufacturer);
    }
    public TV(int size, String manufacturer) {
        this.size = size; this.manufacturer = manufacturer;
        System.out.println(size + "인치 " + manufacturer);
    }
}
```

⑵ new TV(); 와 new TV("삼성");를 실행하면 다음과 같이 각각 실행된다.

```
32인치 LG
32인치 삼성
```

⑶ TV b = new TV(65, "삼성");

⑷ 생성자 코드가 중복 없이 간결해지고, 초기화 코드를 하나의 생성자로 몰아 놓을 수 있다.

7. 컴파일 오류가 발생하는 부분은 다음 코드이며,

```
aPerson.age = 17;
```

오류 메시지는 다음과 같다.

```
The field Person.age is not visible
```

오류가 발생하는 이유는 age가 private으로 선언되어 있기 때문에, Person 클래스 바깥의 main() 메소드에서는 접근할 수 없기 때문이다.
오류를 수정하기 위해 age를 public으로 선언해서는 안 되며, Person 클래스에 이 필드를 접근할 수 있는 public 메소드를 만들어 둔다. 그러면 main()에서 이 메소드를 이용하여 age에 접근할 수 있다. 코드는 다음과 같다.

```
class Person {
    private int age;
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
}
public class Example {
    public static void main (String args[]) {
        Person aPerson = new Person();
        aPerson.setAge(17);
    }
}
```

이 글을 열심히 읽는 독자를 위해서 한 가지 덤으로 중요한 설명을 해보자. 앞의 정답처럼 Person 클래스를 만들어도, main()에서 setAge()를 통해서 age를 마음대로 바꿀 수 있는데 뭐 하러 age를 private으로 두고 굳이 복잡하게 setAge(), getAge()를 만드는가? 하는 의문을 가질 수 있다. 하지만, setAge()를 다음과 같이 만들어 보자.

```
public void setAge(int age) {
    if(age < 0)
        return;
    this.age = age;
}
```

setAge()를 이렇게 만들면 Person 클래스의 age 필드를 음수가 되지 않도록 유지할 수 있지만, setAge() 없이 age를 public으로 공개해버리면, main() 메소드에서 다음과 같이 age를 음수로 설정하는 잘못을 저질러도 막을 수 없게 된다.

```
public static void main (String args[]) {
    Person aPerson = new Person();
    aPerson.age = -20;
}
```

정리하면, 클래스의 주요 필드는 private으로 해두고, public 속성의 set/get 메소드를 별도로 만들어 이 메소드를 통해서만 필드를 접근하게 하여 필드의 무결성을 유지하는 것이 좋은 객체지향프로그래밍이다.

8. (1) 생성자를 이용하여 수정하면 다음과 같다.

```
class Power {
    private int kick;
    private int punch;
    public Power(int kick, int punch) {
        this.kick = kick;
        this.punch = punch;
    }
}
public class Example {
    public static void main (String args[]) {
        Power robot = new Power(10, 20);
    }
}
```

⑵ 생성자 대신 set() 메소드를 추가하여 수정하면 다음과 같다.

```
class Power {
    private int kick;
    private int punch;
    public void set(int kick, int punch) {
        this.kick = kick;
        this.punch = punch;
    }
}
public class Example {
    public static void main (String args[]) {
        Power robot = new Power();
        robot.set(10,  20);
    }
}
```

9.

자바에서는 객체를 임의로 소멸시킬 수 없으며, 이것은 개발자에게 매우 다행한 일이다. 참조하는 레퍼런스가 하나도 없는 객체를 가비지라고 판단하고, 이를 가용 메모리로 자동 수집하는 가비지 컬렉션을 진행시킨다. 응용프로그램에서 자바 플랫폼에 이 과정을 지시하고자 하면 System.gc() 코드를 호출하면 된다.

10. ⑴ s = new String("Hello"+n); 라인을 실행할 때 가비지가 발생한다. for 문에서 n이 0일 때는 가비지가 발생하지 않지만, n이 1일 때 s는 새로운 String 객체를 가리키게 되어 이전에 생성된 String 객체는 가비지가 된다. 가비지 발생은 n이 1부터 9까지 반복된다.

⑵ 가비지가 발생하지 않는다. 처음 세 라인이 실행되면 a, b, c 모두 new String("aa")에 의해 생성된 String 객체를 가리키므로 a와 b가 null이 되어도 레퍼런스 c가 String 객체를 여전히 가리키고 있기 때문이다.

11. ④ static int g() { return getB(); }
static 메소드에서 non-static 멤버를 접근할 수 없다.

12. ① StaticSample.x = 5;
non-static 멤버 x는 클래스 이름으로 접근할 수 없다.

13. 코드에는 틀린 부분이 있다. main() 메소드 내에서 int sum = f(2, 4);의 호출은 잘못되었다. main()은 static 타입이므로 f()를 호출하려면 f()도 static 타입이어야 한다. f() 메소드를 다음과 같이 고쳐야 한다.

```
static public int f(int a, int b)  { return a + b; }
```

14. 잘못된 것 1) x가 final이므로 x의 값을 수정하는 x++가 허용되지 않는다.
잘못된 것 2) Rect가 final이므로, SpeicialRect에서 상속받을 수 없다. 그러므로 class SpecialRect extends Rect는 틀렸다.
잘못된 것 3) Rect의 f()가 final이므로 오버라이딩이 안 된다. 그러므로 SpecialRect에서 f()를 오버라이딩하면 안 되며 SpecialRect 클래스에 public void f() {…} 메소드는 작성할 수 없다.

15. new 연산자를 이용하여 시스템으로부터 할당받아 사용하다 더 이상 사용하지 않는 객체나 배열 메모리를 가비지라 한다. 가비지가 많아지면 상대적으로 자바 가상 기계에서 응용프로그램에게 할당해줄 수 있는 가용 메모리의 양이 줄어들어 자바 응용프로그램의 실행에 영향을 줄 수 있으므로 자바 가상 기계는 가용 공간이 일정 크기 이하로 줄어들게 되면 자동으로 가비지를 회수하여 가용 메모리 공간을 늘린다. 이러한 가비지 컬렉션 때문에 개발자는 할당받은 메모리를 반환하는 코딩 부담을 덜게 된다.

16.

	디폴트	public	protected	private
같은 패키지 클래스	○	○	○	×
다른 패키지 클래스	×	○	×	×

5장 연습문제

이론문제

1. (1) 객체 objA의 멤버들은 총 2개로서 다음과 같다.

```
private int a;
public void set(int a) { this.a = a; }
```

(2) 객체 objB의 멤버들을 총 4개로서 다음과 같다.

```
private int a;
public void set(int a) { this.a = a; }
protected int b, c;
```

(3) 객체 objC의 멤버들은 총 6개로서 다음과 같다.

```
private int a;
public void set(int a) { this.a = a; }
protected int b, c;
public int d, e;
```

(4) a = 1; // ① 라인에서 오류가 발생한다. a는 클래스 A의 private 멤버이므로 상속받은 클래스 D에서 접근할 수 없기 때문이다. 하지만, protected 멤버는 마음대로 접근할 수 있다.

2. ① Object

3. 독자마다 조금씩 다르게 할 수도 있겠지만, 현재와 미래에 확장성으로 고려하여 다음과 같이 작성하였다.

```
class Pen { // 모든 펜의 공통 속성
    private int amount; // 남은 량
    public int getAmount() { return amount; }
    public void setAmount(int amount) { this.amount = amount; }
}
class SharpPencil extends Pen {
```

```
    private int width; // 펜의 굵기
}
class BallPen extends Pen {
    private String color;
    public String getColor() { return color; }
    public void setColor(String color ) { this.color = color; }
}
class FountainPen extends BallPen {
    public void refill(int n) { setAmount(n); }
}
```

다음과 같이 작성할 수도 있다.

```
class Pen {
    private int amount;
    public int getAmount() { return amount; }
    public void setAmount(int amount) { this.amount = amount; }
}

class SharpPencil extends Pen {
    private int width; // 펜의 굵기
}

class ColorPen extends Pen {
    private String color;
    public String getColor() { return color; }
    public void setColor(String color ) { this.color = color; }
}

class BallPen extends ColorPen {

}

class FountainPen extends ColorPen {
    public void refill(int n) { setAmount(n); }
}
```

4.

자바에서 상속받는 클래스를 서브 클래스라고 부르며, extends 키워드를 이용하여 상속을 선언한다. 상속받은 클래스에서 슈퍼 클래스의 멤버를 접근할 때 super 키워드를 이용한다. 한편, 객체가 어떤 클래스의 타입 인지 알아내기 위해서는 instanceof 연산자를 이용하며, 인터페이스는 클래스와 달리 interface 키워드를 이용하여 선언한다.

5. ②. protected 멤버는 다른 패키지의 서브 클래스에서도 접근 가능하다.

6.

```
class TV {
    private int size;
    public TV(int n) {size = n;}
}
class ColorTV extends TV {
    private int colors;
    public ColorTV(int colors, int size) {
         super(size);
        this.colors = colors;
    }
}
```

7. 다음과 같이 출력된다.

```
A
B:11
```

8. 클래스 A의 생성자를 protected로 사용해도 무관하므로 다음은 오류가 아니다.

```
protected A(int i) { a = i; }
```

잘못된 부분은 클래스 B의 다음 생성자에 있다.

```
public B() { b = 0; }
```

자바 컴파일러는 이 문장을 컴파일할 때, 클래스 A의 매개변수 없는 생성자를 호출하도록 컴파일하는데 클래스 A에는 매개변수 없는 기본 생성자가 만들어져 있지 않다. 그래서 다음과 같은 컴파일 오류 메시지가 출력된다.

`Implicit super constructor A() is undefined. Must explicitly invoke another constructor`

오류를 수정하는 방법에는 2가지가 있다.
첫째, 클래스 A에 다음 생성자를 추가하는 방법이다.

```
public A() { }
```

둘째, 클래스 A에 문제를 삼는 것은 바람직하지 않다. 상속받는 사람이 클래스 A를의 의미를 훼손하지 않고 상속해야하는 것이 정상이다. 그러므로 클래스를 B의 생성자를 다음과 같이 수정한다. 이것이 정답이다.

```
public B() {
    super(0);  // 클래스 A의 A(int i)의 생성자 호출.
    b = 0;
}
```

9. ①. 잘못됨. 추상 메소드 `void f()`를 `abstract void f();`로 수정하여야 한다.
②. 문제 없음. 추상 메소드가 없는 클래스로 추상 클래스로 선언할 수 있다.
③. 잘못됨. 추상 클래스 B를 상속받고 추상 메소드를 오버라이딩하지 않으면 클래스 C도 추상 클래스가 됨. 그러므로 클래스 C를 다음과 같이 수정하여야 한다.

```
abstract class C extends B {
}
```

④. 잘못됨. 추상 클래스 B의 추상 메소드는 리턴 타입이 int이지만, 상속받은 클래스 C에서는 void 타입의 메소드를 f()를 구현하였기 때문에 오버라이딩이 실패하였음. 다음과 같이 수정하여야 한다.

```
class C extends B {
    int f() { System.out.println("~"); return 0;}
}
```

10.

```
abstract class OddDetector  {
    protected int n;
    public OddDetector (int n) {
        this.n = n;
    }
    public abstract boolean isOdd();
}
public class B extends OddDetector  {
    public B(int n) {
        super(n);
    }
    public boolean isOdd() { // isOdd() 메소드 오버라이딩
        if(n%2 == 0) return false;
        else return true;
    }
    public static void main(String [] args) {
        B b = new B(10);
        System.out.println(b.isOdd());
    }
}
```

11. ① ② B b = new C(); ③ A a = new D();

(2)

```
true
false
```

(3)

```
true
true
```

12. (1) Circle

(2) draw();

(3) super.draw();

13. ⑴ 추상 클래스의 객체는 생성할 수 없다. 그러므로 다음 두 경우가 오류이다.
② `Shape s = new Shape();` ④ `Circle c = new Circle(10);`

⑵ `Circle` 클래스에 2군데를 수정해야 한다.

```
class Circle extends Shape { // abstract 삭제
    private int radius;
    public Circle(int radius) { this.radius = radius; }
    double getArea() { return 3.14*radius*radius; }

    // draw() 오버라이딩
    public void draw() { System.out.println("반지름=" +  radius); }
}
```

14. ④. 자바에서 다형성은 모호한(ambiguous) 문제를 일으키므로 사용하지 않는 것이 바람직하다.

15. ②. 인터페이스는 클래스와 같이 멤버 변수(필드)의 선언이 가능하다.

16.

```
interface Device {
    void on();
    void off();
}
public class TV implements Device {
    public void on() { // Device의 on() 메소드 구현
        System.out.println("켜졌습니다.");
    }
    public void off() { // Device의 off() 메소드 구현
        System.out.println("종료합니다.");
    }
    public void watch() { // 새로운 메소드 작성
        System.out.println("방송중입니다.");
    }
    public static void main(String [] args) {
        TV myTV = new TV();
        myTV.on();
        myTV.watch();
        myTV.off();
    }
}
```

6장 연습문제

이론문제

1. (1) import 문은 다른 패키지에 있는 클래스를 사용할 때 코드의 서두에 선언하는 것으로, 컴파일러에게 그 클래스의 경로명을 알려주는 문이다.

   (2) import java.util.Random;은 Random 클래스가 java.util 패키지에 있음을 컴파일러에게 알려주는 문이다. 자바 소스 프로그램에서 Random의 이름을 사용하면, 컴파일러가 Random 클래스의 경로명을 찾을 때, import 문을 참조하여 찾게 한다. import java.util.*;는 자바 프로그램 내에서 사용하는 클래스들의 경로명을 찾을 때, java.util 패키지에서도 확인할 것을 컴파일러에게 지시하는 문이다. 만일 자바 소스 프로그램에서 Random 클래스를 사용하면, 컴파일러는 Random 클래스가 어느 패키지에 있는지 찾기 위해 java.util 패키지에서 확인해본다. import java.util.*;는 import java.util.Random; import java.util.Vector; 등과 같이 여러 import 문을 줄여 사용할 때 유용하다.

   (3) import 문을 사용하지 않고도 자바 프로그램을 작성할 수 있다. 자바 프로그램을 작성할 때, 모든 클래스의 이름에 완전 경로명을 사용하면 된다. 예를 들면 java.util.Random r = new java.util.Random(); 과 같이 완전 경로명으로 작성하는 것이다. 하지만, 프로그램의 코드가 길어지고 복잡해져 가독성이 떨어지는 단점이 있어 import 문을 사용하는 것이 효과적이다.

   (4) java.lang 패키지에 속한 클래스들은 import 없이도 사용할 수 있다. java.lang 패키지에 속한 클래스들로는 Object, String, Math, System 등이 있다.

2. ④. 자바 JDK는 수많은 클래스들을 제공하는데, 서로 관련 있는 클래스들을 하나의 패키지로 구성하는 식으로 수 많은 패키지에 분산 저장하여 제공한다.

3. import 문을 사용하지 않도록 Example 클래스를 다시 작성하면 다음과 같다.

```
public class Example {
    public static void main(String[] args) {
        java.util.StringTokenizer st = new java.util.StringTokenizer("a=3,b=5,c=6", ",");
        while (st.hasMoreTokens())
            System.out.println(st.nextToken());
    }
}
```

4. 여러 가지로 작성할 수 있는데, 첫 번째 다음과 같이 작성할 수 있다.

```
import java.util.StringTokenizer;
public class Example {
    public static void main(String[] args) {
        int sum = 0;
        StringTokenizer st = new StringTokenizer("a=3,b=5,c=6", ",=");
        while (st.hasMoreTokens()) {
            String token = st.nextToken();
            System.out.println(token);
            int n = 0;
            try {
                n = Integer.parseInt(token);
            }
            catch(NumberFormatException e) {
                // 정수가 아닌 경우 아무것도 하지 않고 그냥 넘어 간다.
            }
            sum += n;
        }
        System.out.println("합은 " + sum);
    }
}
```

좀 다르게 작성하면 다음과 같다.

```
import java.util.StringTokenizer;
public class Example {
    public static void main(String[] args) {
        int sum = 0;
        int n = 1;
        StringTokenizer st = new StringTokenizer("a=3,b=5,c=6", ",=");
        while (st.hasMoreTokens()) {
            String token = st.nextToken();
            System.out.println(token);
            if(n%2 == 0) { // 짝수번째 토컨에 대해서만 정수로 변환하여 합하기
                sum += Integer.parseInt(token);
            }
            n++;
        }
        System.out.println("합은 " + sum);
    }
}
```

5. (1)

```
package device;
public class TV {
    private int size;
    public TV(int size) { this.size = size; }
}
```

(2)

```
package app;
import device.TV;
public class Home {
    public Home() { TV myTv = new TV(65); }
}
```

package app;와 import device.TV;의 순서가 다르면 틀린 답이다. 반드시 package 선언이 먼저 와야 한다.

(3) TV 클래스를 컴파일한 TV.class 파일의 경로명은 device.TV.class이고, Home 클래스를 컴파일한 Home.class 파일의 경로명은 app.Home.class이다.

6. (1) String s1 = Integer.toString(20);
(2) double d = Double.parseDouble("35.9");
(3) boolean b = Boolean.parseBoolean("true");
(4) String s2 = Integer.toBinaryString(30);
(5) String c = Character.toString('c');

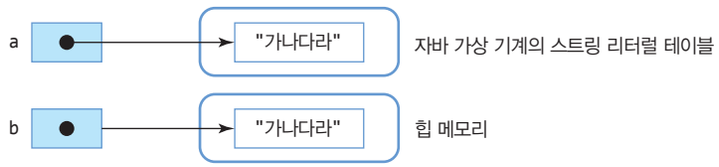
7. 다음 코드를 실행하면,

```
String a = "가나다라";
System.out.println(a == "가나다라");
String b = new String(a);
System.out.println(a == b);
```

다음과 같이 출력된다.

```
true
false
```

설명을 위해 다음 그림을 참고하라.



"가나다라" 문자열은 자바 가상 기계의 스트링 리터럴 테이블에 저장되고, String a의 레퍼런스 a는 리터럴 테이블을 가리키므로, "가나다라" 문자열의 레퍼런스와 레퍼런스 a의 값은 일치한다. 그러므로 a == "가나다라"의 비교 연산은 true이다. 하지만, new String(a)에 의해 생성되는 객체는 힙에 생성되므로 String b는 힙 영역을 가리킨다. 그러므로 a == b의 연산에서 a와 b의 레퍼런스 값은 서로 달라 연산 결과는 false가 된다.

8. 다음 코드 실행된 후 a, b, c 문자열은 무엇인지 알아보자.

```
String a = new String("     Oh, Happy   ");
String b = a.trim();
String c = b.concat(" Day.");
```

a.trim() 후에도 문자열 a는 변하지 않는다. 다만 trim()은 문자열 앞뒤의 공백문자를 제거한 새로운 문자열을 리턴하므로 a와 b는 다음과 같이 된다.
문자열 a : "   Oh, Happy   "
문자열 b : "Oh, Happy"

그리고 b.concat() 후에도 문자열 b는 변하지 않고, 문자열을 연결한 새로운 문자열을 리턴한다. 그러므로 문자열 c는 다음과 같다.
문자열 c : "Oh, Happy Day."

String 클래스의 메소드는 현재 문자열을 수정하지 않는다. 이것은 String 객체의 특징이다.

9. (1) b
(2) c, e

10. (1) Double PI = 3.14; // 3.14를 Double(3.14)로 자동 박싱
(2) double pi = PI; // PI를 PI.doubleValue();로 자동 언박싱

(3) System.out.println(3 + Integer.valueOf(5));
// Integer.valueOf(5)는 Integer.valueOf(5).intValue()로 자동 언박싱되어
// 실행 중에 5가 된다.

(4) if('c' == Character.valueOf('c'))
// Character.valueOf('c')가 Character.valueOf('c').charValue()로 자동 언박싱되어 실행 중에 'c'가 된다.

11. (1) Math 클래스를 활용하여 100~255까지의 랜덤 정수 발생

```
for(int i=0; i<10; i++)
    System.out.println((int)(Math.random()*156 + 100));
```

(2) Random 클래스를 난수를 발생시키도록 코드 전체를 재작성하면 다음과 같다.

```
import java.util.Random;
public class Example {
    public static void main(String[] args) {
        Random r = new Random();
        for(int i=0; i<10; i++) {
            System.out.print(r.nextInt(156) + 100);
            System.out.print(" ");
        }
    }
}
```

12.

```
Calendar date = Calendar.getInstance(); // Calendar 객체 생성
date.clear(); // 현재 Calendar 객체에 저장된 정보를 모두 지운다.
date.set(Calendar.YEAR, 2020); // Calendar 객체에 2020년의 년도 저장
date.set(Calendar.MONTH, 11); // Calendar 객체에 12월의 달 저장
date.set(Calendar.DAY_OF_MONTH, 25); // Calendar 객체에 25일의 날짜 저장
System.out.println("약속 날짜는 " + date.get(Calendar.YEAR) + "년 " +
 (date.get(Calendar.MONTH) + 1) + "월 " + date.get(Calendar.DAY_OF_MONTH)+ 일");
```

7장 연습문제

이론문제

1. ③. int와 같은 기본 타입의 값은 Wrapper 클래스를 이용하여 객체로 만들어 저장하면 된다.

2.

> 클래스, 인터페이스, 메소드를 특정 타입에 종속되지 않게 일반화시켜 작성하는 기술을 제네릭이라 한다. Vector<E>에서 E는 제네릭 타입 혹은 타입 매개변수라고 부른다. E에 Integer 등과 같이 객체 타입을 지정하여 특정 타입의 데이터만 다루도록 만드는 작업을 구체화라고 한다. 제네릭 기능은 C++에서 먼저 만들어졌으며 C++에서는 템플릿이라고 부른다.

3. ②. 벡터 v의 초기 크기가 30이다. 30개 이상 저장할 수 있고 벡터는 스스로 저장 공간을 늘린다.

4. ⑴ `Vector<String> sv = new Vector<String>();`
   ⑵ `HashMap<String, Double> h = new HashMap<String, Double>();`
   ⑶ `ArrayList<Person> pa = new ArrayList<Person>(10);`
   ⑷ `HashMap<String, Integer> pop = new HashMap<String, Integer>();`

5. ① `Stack<String> ss;`
   다른 문항이 틀린 이유는 다음과 같다.
   ②에서는 E에 구체적인 타입을 지정하여야 사용할 수 있다.
   ③에서는 HashMap에서는 두 개의 타입 매개변수가 필요한데 String 타입 하나만 사용하였기 때문이다.
   ④에서 Set은 인터페이스이므로 `new Set<Integer>()`와 같이 객체를 생성할 수 없다.

6.

```
v.add(3.14); // 3.14가 new Double(3.14)로 바뀌는 자동 박싱 발생
double d = v.get(0); // v.get(0)이 v.get(0).doubleValue()로 바뀌는 자동 언박싱 발생
```

7. 최종적으로 출력되는 벡터의 용량은 12이다. 벡터의 초기 용량은 3이며 루프가 4번째 돌 때 벡터의 공간이 부족하다. 이때 벡터는 용량을 2배 증가시켜 용량이 6이 된다. 다시 7번째 루프를 돌 때 용량이 12가 되며, 10번 루프를 돌았을 때의 용량은 여전이 12이다. 코드를 아래와 같이 고쳐서 실행해보면 더욱 분명해진다.

```
Vector<Integer> v = new Vector<Integer>(3);
for(int i=0; i<10; i++) {
    v.add(i);
    System.out.println(v.capacity()); // 현재 용량 출력
}
```

8.

```
ArrayList<String> a = new ArrayList<String>(10); // 초기 용량이 10인 ArrayList 생성
a.add("Java");                    // a의 맨 끝에 "Java" 삽입
a.add(0, "C++a");                 // a의 맨 앞에  "C++" 삽입
System.out.println(a.size());     // a에 현재 삽입된 개수 출력
a.remove(a.size()-1);             // a의 마지막에 있는 문자열 삭제
```

9.

```
Vector<Integer> v = new Vector<Integer>();
for(int i=0; i<10; i++) v.add(i); // 10개의 정수 저장

Iterator<Integer> it = v.iterator(); // Iterator 레퍼런스 얻기
while(it.hasNext()) { // it가 가리키는 객체가 있는 동안 루프
    int n = it.next(); // 하나씩 알아내기
    System.out.print(n + " "); // 출력
}
```

10. 이 문제는 HashMap 객체를 메소드의 매개변수로 어떻게 전달받는지를 묻는 문제이다. func()와 이를 응용하는 코드를 보면 다음과 같다.

```
import java.util.HashMap;
public class Example {
    public static void main(String[] args) {
        HashMap<String, Integer> h = new HashMap<String, Integer>();
        h.put("a", 10);
        h.put("b", 20);
```

```
        System.out.println(func(h)); // 해시맵 map에 들어 있는 원소 개수 출력
    }
    public static int func(HashMap<String, Integer> map) {
        return map.size();
    }
}
```

11. (1) JGeneric의 타입 매개변수는 1개이며 W이다.
(2), (3), (4), (5), (6)은 아래 코드와 같다.

```
class JGeneric<W> {
    private W x, y;
    public JGeneric(W x, W y) {
        this.x = x; this.y = y;
    }
    public W first() { return x; }  // (3) 여러 줄로 작성 가능
    public W second() { return y; }  // (4) 여러 줄로 작성 가능
    public boolean equal() { return x.equals(y); }  // (5) 여러 줄로 작성 가능
}

public class Example {
    public static void main(String[] args) {
        JGeneric<String> js = new JGeneric<String>("hello", "hellu"); // (2)
        // (6) 활용 예
        System.out.println(js.first());
        System.out.println(js.second());
        System.out.println(js.equal());

    }
}
```

```
hello
hellu
false
```

12. (1) JClass의 take() 메소드는 문자열 배열과 인덱스를 받아 인덱스에 있는 문자열을 리턴하는 메소드이다.
(2) take() 메소드를 일반화시키는 제네릭 메소드는 다음과 같이 정의한다. 첫 번째 매개변수로는 어떤 타입이든지 배열로 받고, 두 번째 매개변수는 항상 int

타입의 인덱스 값을 받아 배열의 원소를 리턴하도록 제네릭 메소드로 작성한다. JGenClass는 다음과 같다.

```
class JGenClass {
    static <T> T take(T s[], int index) {
        if (index > s.length) {
            System.out.println("인덱스가 범위를 벗어났습니다.");
            return null;
        }
        return s[index];
    }
}
```

(3) JGenClass를 활용하는 코드는 다음과 같다.

```
class JGenClass {
    static <T> T take(T s[], int index) {
        if (index > s.length) {
            System.out.println("인덱스가 범위를 벗어났습니다.");
            return null;
        }
        return s[index];
    }
}

public class Example {
    public static void main(String[] args) {
        String [] text = { "Java", "C++", "c#" };
        System.out.println(JGenClass.take(text, 1)); // 문자열 배열에 대해

        Integer [] n = { 0,1,2,3,4,5 }; // int []로 하면 안 된다.
        System.out.println(JGenClass.take(n, 1)); // 정수 배열에 대해
    }
}
```

8장 연습문제

이론문제

1. ②. 자바에서 스트림은 다른 스트림과 연결하여 사용함으로써 다양한 데이터의 입출력을 가능하게 한다.

2. (1) 음악 파일 연주 – 음악 파일은 바이너리 파일이므로, 파일 입력 바이트 스트림 FileInputStream 필요
   (2) PPT 파일 복사 – PPT 파일은 바이너리 파일이고 읽고 쓰는 작업을 해야 하므로 파일 입력 바이트 스트림 FileInputStream과 파일 출력 바이트 스트림 FileOutputStream 필요
   (3) 영어와 한글이 기록된 사전 읽기 – 영어 단어와 한글 단어는 모두 텍스트 정보이므로 이들이 기록된 파일을 읽기 위해서는 파일 문자 입력 스트림 FileReader 필요
   (4) 선수 이름과 점수를 문자열로 저장 – 선수 이름과 점수를 문자열로 저장하면 텍스트 파일이 만들어지는데 이를 위해서는 파일 출력 문자 스트림 FileWriter 필요

3. ④ FileInputStream

4. 파일을 찾을 수 없는 예외가 발생할 수 있다. 그러므로 다음과 같이 FileNotFoundException 예외 처리 블록을 구성한다.

```
FileInputStream fin=null;
try {
    fin = new FileInputStream("song.mp3");
} catch(FileNotFoundException e) {
    System.out.println("파일을 찾을 수 없습니다.");
}
```

5. ① File

6. ① 파일 읽고 쓰기

7. (1) true
(2) "c:\\windows"
(3) "c:\\windows\\system.ini"
(4) "system.ini"
(5) File file = new File("c:\\windows\\", "system.ini"); 또는
File file = new File("c:\\windows", "system.ini");

8. ③. 바이트 스트림 입출력 시 문자 집합을 고려할 필요 없다. 문자 단위로 입출력하는 것이 아니라 파일에 있는 바이너리 정보 그대로 읽고 쓰는 것이기 때문이다.

9.

```
FileInputStream fin = null;
try {
    fin = new FileInputStream("c:\\temp\\test.txt");
    int c;
    while(true) {
        c = fin.read(); // 파일에서 한 바이트 읽기
        if (c == -1)  break; // 파일 끝까지 읽었음
        System.out.print((char)c);
    }
    fin.close(); // 파일 입력 스트림 닫기
} catch (FileNotFoundException e1) {
    System.out.println("파일을 찾을 수 없습니다.");
} catch (IOException e) {
    System.out.println("입출력 오류가 발생했습니다.");
}
```

10.

```
BufferedOutputStream bout = new BufferedOutputStream(System.out, 50);
File f = new File("c:\\temp\\sample.txt");
FileReader fin = new FileReader(f);
int c;
while ((c = fin.read()) != -1) {
    bout.write((char)c);
}
fin.close();
bout.close();
```

9장 연습문제

이론문제

1. AWT 컴포넌트는 중량 컴포넌트이며, Swing 컴포넌트는 경량 컴포넌트이다. 중량 컴포넌트는 운영체제(다른 말로 native)가 가진 GUI 자원을 할당받고 이를 이용하여 화면에 출력되는 컴포넌트로서, 운영체제 자원을 소모하여 운영체제에 부담을 준다. 이에 반해 경량 컴포넌트란 운영체제의 자원을 활용하지 않고 그려지는 GUI 컴포넌트이다. 그러므로 AWT 컴포넌트는 운영체제에 따라 서로 다른 모양을 출력될 가능성이 높지만, 경량 컴포넌트는 운영체제와 관계없이 동일한 모양으로 출력된다. 또한 AWT 컴포넌트는 운영체제가 직접 화면에 그리기 때문에 그려지는 속도가 상대적으로 빠르다.

2. 스윙 컴포넌트들은 AWT보다 모양이 예쁘고 다양하며 운영체제에 부담을 덜 주기 때문이다.

3. ④ Button

4. ① JFrame, Font, Color, Graphics는 컴포넌트가 아니고 AWT 패키지에 포함되어 GUI에 사용되는 객체들이다.

5. ④. 컴포넌트들은 컨테이너 없이도 출력된다.

6. 배치관리자에 대해 잘못 말한 것은 다음과 같다.
   ②. 배치관리자는 컴포넌트의 크기와 위치를 조절한다.
   ③. 컨테이너가 생성될 때 디폴트 배치관리자을 가진다.
   ④. 한 컨테이너는 오직 한 개의 배치관리자만 가질 수 있다.
   ⑦. 컨테이너가 배치관리자를 가지지 않도록 할 수 있다.

7.

```
import javax.swing.*;  // 이곳에 필요한 import 문을 삽입하라.
public class MyFrame extends JFrame {
    public MyFrame() {
        setTitle("hello"); // 프레임 타이틀로 "hello" 문자열 출력
```

```
        setSize(200,300); // 프레임 크기를 200x300으로 설정
        setVisible(true); // 프레임 화면 출력
    }
    public static void main(String [] args) {
        MyFrame frame = new MyFrame();
    }
}
```

8.

```
import javax.swing.*; // 이곳에 필요한 import 문을 삽입하라.
import java.awt.*;    // 이곳에 필요한 import 문을 삽입하라.
public class MyFrame extends JFrame {
    public MyFrame() {
        Container c = getContentPane() ; // 컨텐트팬 알아내기
        c.setLayout(new FlowLayout()) ; // 컨텐트팬에 FlowLayout 배치관리자 설정
        c.setBackground(Color.YELLOW); // 컨텐트팬 배경색을 노란색으로 설정
        c.add(new JButton("click")); // 컨텐트팬에 "click" 버튼 달기
        setSize(300,300);
        setVisible(true);
    }
}
```

9. (1) c.setLayout(new BorderLayout(3, 4));
   (2) c.setLayout(new FlowLayout(FlowLayout.RIGHT, 5, 6));
   (3) c.setLayout(new GridLayout(5, 2, 7, 8));
   (4) c.setLayout(null);

10.

```
import java.awt.Container; // 이곳에 필요한 import 문을 삽입하라.
import javax.swing.*; // 이곳에 필요한 import 문을 삽입하라.

// 이곳에 필요한 import 문을 삽입하라.
public class MyFrame extends JFrame {
    public MyFrame() {
        Container c = getContentPane();     // 컨텐트팬 알아내기
        c.setLayout(null);                  // 컨텐트팬에 배치관리자 제거
        JButton b = new JButton("click");   // "click" 문자열의 버튼 컴포넌트 생성
        b.setSize(100, 30);                 // 버튼의 크기를 100x30으로 조절
```

```
        b.setLocation(50, 70);              // 버튼의 위치를 (50, 70)으로 조절
        c.add(b);                           // 컨텐트팬에 버튼 달기
        setSize(300,300);
        setVisible(true);
    }
    public static void main(String [] args) {
        new MyFrame();
    }
}
```

10장 연습문제

이론문제

1. ③. 키 이벤트를 처리하는 도중 마우스 이벤트가 발생하면, 현재 이벤트를 모두 처리한 뒤 다음 이벤트를 처리한다. 이벤트는 발생 순서대로 처리된다.

2. ④ 마우스 드래깅 길이

3. 익명 클래스를 이용하여 다시 작성하면 다음과 같다.

```
JButton btn = new JButton("Hello");

btn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        System.out.println("Click");
    }
});
```

4. 익명 클래스를 이용하여 다시 작성하면 다음과 같다.

```
JButton btn = new JButton("Hello");

btn.addKeyListener(new KeyAdapter() {
    public void keyReleased(KeyEvent e) {
        System.out.println("Key Released");
    }
});
```

5. (1) ActionAdapter는 존재하지 않으며 implements ActionListener로 수정하여야 한다.

```
class MyActionListener extends ActionAdapter { // implements ActionListener로 수정
    public void actionPerformed(ActionEvent e) {
        System.out.println("Click");
    }
}
```

⑵ MouseListener를 implements하면 나머지 4개 mouseReleased(), mouseClicked(), mouseEntered(), mouseExited()를 모두 구현하여야 한다. 그러므로 extends MouseAdapter로 수정하여야 한다.

```
class MyMouseListener implements MouseListener { // extends MouseAdapter로 수정
    public void mousePressed(MouseEvent e) {
        System.out.println("Mouse Pressed");
    }
}
```

⑶ Key 이벤트가 발생하면 KeyEvent 객체가 생성되고 메소드로 넘어온다. 그러므로 ActionEvent가 아니라 KeyEvent로 수정하여야 한다.

```
class MyKeyListener extends KeyAdapter {
    public void keyTyped(ActionEvent e) { // KeyEvent로 수정
        System.out.println("Key Pressed");
    }
}
```

6. ① 3 2 1이 순서대로 출력된다.
동일한 이벤트에 대해 여러 개의 리스너를 등록할 수 있으며, 등록된 리스너를 등록된 순서의 반대로 순으로 실행되며, 모두 실행된다. 정답을 확인하기 위해 다음 코드를 작성해보았다.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MyFrame extends JFrame {
    public MyFrame() {
        Container c = getContentPane();
        c.setLayout(new FlowLayout());
        JButton btn = new JButton("Hello");
        c.add(btn);

        btn.addActionListener(new MyActionListener("1"));
        btn.addActionListener(new MyActionListener("2"));
        btn.addActionListener(new MyActionListener("3"));

        setSize(300,300);
```

```
        setVisible(true);
    }
    public static void main(String [] args) {
        new MyFrame();
    }
}

class MyActionListener implements ActionListener {
    private String msg;
    public MyActionListener(String msg) { this.msg = msg; }
    public void actionPerformed(ActionEvent e) { System.out.print(msg + " "); }
}
```

7. ① ItemListener

8. ③ component.requestFocus();

9. <Alt>, <Tab>, <Delete>, <Shift>, <Help>

10. (1)

```
public void keyPressed(KeyEvent e) {
    if(e.getKeyChar() == 'a') {
        System.exit(0);
    }
}
          }
```

(2)

```
public void keyPressed(KeyEvent e) {
    if(e.getKeyCode() == KeyEvent.VK_ALT) {
        System.exit(0);
    }
}
```

이 문제의 테스트를 위한 코드는 다음과 같다.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MyFrame extends JFrame {
    public MyFrame() {
        Container c = getContentPane();
        c.addKeyListener(new KeyAdapter() {
            public void keyPressed(KeyEvent e) {
                if(e.getKeyChar() == 'a') {
                    System.exit(0);
                }
            }
        });

        c.addKeyListener(new KeyAdapter() {
            public void keyPressed(KeyEvent e) {
                if(e.getKeyCode() == KeyEvent.VK_ALT) {
                    System.exit(0);
                }
            }
        });

        setSize(300,300);
        setVisible(true);

        c.setFocusable(true);
        c.requestFocus();
    }
    public static void main(String [] args) {
        new MyFrame();
    }
}
```

11. (1) a 키는 유니코드 키이므로 keyPressed(), keyTyped(), keyReleased() 순으로 호출된다.

(2) <Esc> 키는 유니코드 키가 아니므로 keyPressed(), keyReleased() 순으로 호출되며, keyTyped()는 호출되지 않는다.

12. (1) 1을 출력한다. 그 이유는 다음과 같다. 마우스를 누르면 contentPane.addMouseListener(new MyMouseListener());로 등록한 마우스 이벤트 리스너 객체인 new MyMouseListener() 객체의 mousePressed()가 실행되어 count는 1이 되고 마우스를 놓으면 mouseReleased()가 실행되어 count 값 1을 출력한다.

(2) 1을 출력한다. 그 이유는 다음과 같다. 마우스를 누르면 contentPane.addMouseListener(new MyMouseListener());로 등록한 new MyMouseListener() 객체의 mousePressed()가 실행되어 count는 1이 된다. 그러나 마우스를 드래그하면 contentPane.addMouseMotionListener(new MyMouseListener());로 등록한 new MyMouseListener() 객체의 mouseDragged()가 실행되어 0으로 초기화된 count 값을 10으로 증가시킨다. 마우스를 놓을 때는 처음 마우스 이벤트 리스너의 mouseReleased()가 실행되어 mousePressed()에 의해 객체 속에 있는 count 값 1을 그대로 출력한다. count 멤버는 마우스 리스너 객체와 마우스 모션 리스너 객체에 각각 있기 때문에 mousePressed()와 mouseReleased()는 같은 count를 접근하지만 mouseDragged()는 다른 count를 접근하여 생긴 결과이다.

(3) 이 코드의 경우, new MyMouseListener()로 생성한 객체가 마우스 이벤트 리스너와 마우스모션 이벤트 리스너로 동시에 작동하도록 등록하였다. 그러므로 mousePressed()나, mouseDragged(), mouseReleased() 모두 count 멤버 변수를 공유한다.
이제 (1) 질문에 대한 답은 하면, 1이 출력된다. (2)의 경우 mousePressed()에 의해 count 값이 1이 되고, mouseDragged()가 10번 호출되었다면 count 값은 11이 된다. 그리고 mouseReleased()에 의해 출력되므로 화면에는 11이 출력된다.

13. component.repaint();는 component의 위치나 크기, 색 등에 변화가 생겼으니 다시 그리도록 자바 플랫폼에 요청하는 코드이다. component.revalidate();는 component가 사실상 컨테이너인 경우로, 컨테이너의 자식 컴포넌트들을 다시 배치하도록 지시하는 코드이다. 컨테이너 내부의 자식 컴포넌트가 삭제되거나 새로 추가된 경우 등으로 컨테이너 내부에 변화가 일어난 경우 호출하면 된다.

11장 연습문제

이론문제

1. ③ 컴포넌트가 만들어진 시간

2. ① 이미지나 텍스트 출력

3.

```
ImageIcon icon = new ImageIcon("java.jpg"); // java.jpg 파일을 로딩한다.
JLabel label = new JLabel(); // 빈 JLabel 컴포넌트를 생성한다.
label.setIcon(icon); // 이미지를 레이블에 부착한다.
```

4.

```
c.setVisible(false); // 컴포넌트가 보이지 않도록 만든다.
c.setFont(new Font("Gothic", Font.PLAIN, 20)); // 글자체를 20픽셀의 고딕체로 한다.
c.setEnabled(false); // 컴포넌트가 마우스나 키보드로 입력해도 반응이 없게 한다.
```

5. ④ 슬라이드바를 클릭한 경우

6. ③ JCheckBox 컴포넌트를 마우스로 선택한 경우

7.

```
b.setIcon(new ImageIcon("plain.jpg"));
b.setRolloverIcon(new ImageIcon("over.jpg"));
```

8.

```
class MyItemListener implements ItemListener {
    public void itemStateChanged(ItemEvent e) {
    if(e.getStateChange() == ItemEvent.SELECTED)
        System.out.println("선택되었습니다");
    else
        System.out.println("해제되었습니다");
}
```

9.

```
slider.setValue(slider.getMinimum()+(slider.getMaximum()-slider.getMinimum())/2);
```

10. ButtonGroup 클래스는 라디오버튼들(JRadioButton 객체)을 하나의 그룹으로 만드는 역할이다. 그룹(ButtonGroup)에 속한 라디오버튼들 중에서 하나만 선택된다.

12장 연습문제

이론문제

1. ③. 그래픽 기반 프로그래밍과 컴포넌트 기반 프로그래밍은 함께 사용하면 보다 다양한 GUI를 만들 수 있다.

2. ③

3. ③. Graphics로 선을 그릴 때 선의 두께를 조절할 수 없다.

4. ④

5. AWT 컴포넌트와 스윙 컴포넌트가 화면에 그려지는 과정은 매우 다르다. AWT 컴포넌트의 그리기는 Component 클래스의 paint(), update() 등의 메소드에 의해 지배받지만, 스윙 컴포넌트의 경우 JComponent에 구현된 paint(), paintComponent(), paintChildren() 등의 메소드에 의해 지배받는다. 그러므로 한 컨테이너에 AWT 컴포넌트와 스윙 컴포넌트가 존재하게 되면 AWT 컴포넌트가 화면에 그려지지 않거나 이상하게 그려질 가능성이 있다.

6. ①. paintComponent()가 제일 먼저 호출된다.

7.

⑴ 이미지를 원본 크기로 (10, 20) 위치에 그리는 코드는 비교적 간단하다.

```
g.drawImage(img, 10, 20, this);
```

⑵ 패널에서 상, 하, 좌, 우 10픽셀씩 간격을 두고 그 안에 이미지가 모두 보이도록 그리기 위해서는 시작 좌표는 (10, 10)이며, 이미지의 크기는 폭이 10*2만큼 작게, 높이도 10*2만큼 작게 그려야 한다. 그러므로 다음과 같다.

```
g.drawImage(img, 10, 20, getWidth()-10*2, getHeight()-10*2, this);
```

8. 원을 수직, 수평으로 사등분하였을 때, 오른쪽 상단의 1/4 부분만 보이도록 하기 위해서는 그 영역을 클리핑 영역으로 지정하여야 한다. 이 클리핑 영역의 시작점은 (getWidth()/2, 0)이고, 크기는 폭은 getWidth()/2이며 높이는 getHeight()/2이다. 그러므로 답은 다음과 같다.

```
class MyPanel extends JPanel {
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.setClip(getWidth()/2, 0, getWidth()/2, getHeight()/2);
        g.fillOval(0, 0, getWidth(), getHeight(), this);// 패널에 꽉 찬 원 그리기
    }
}
```

9. ①

10. 컴포넌트의 repaint() 메소드를 호출하면 된다.

13장 연습문제

이론문제

1. 영화 보면서 팝콘 먹기, 전화하면서 문서 작성하기

2. ② 자바에서는 다수의 프로세스를 가진 멀티프로세스를 지원한다.

3. ① run()

4. ① 자바 가상 기계(JVM)

5. (1) 자바 스레드의 우선순위 값의 범위는 1~10 사이
   (2) 우선순위가 높은 스레드를 무조건 먼저 실행시키는 우선순위 기반 정책. 우선순위가 같으면 돌아가면서 실행시키는 라운드로빈 정책
   (3) 스레드 B는 스레드 A가 종료하거나 스레드 A가 sleep()을 호출하여 잠을 자거나 스레드 A가 입출력을 호출하여 block 상태가 되거나 스레드 A가 wait()을 호출하여 누군가 notify()를 불러주기를 무한정 기다릴 때
   (4) 스레드 A, B, C가 실행 준비 상태일 때, 항상 스레드 A의 우선순위가 제일 높기 때문에 자바 가상 기계는 무조건 스레드 A를 선택하여 실행시킨다.
   (5) 스레드 A가 완전히 종료하면, 스레드 B와 C의 우선순위가 같으므로 짧은 시간 동안 나누어 번갈아 실행된다.
   (6) 스레드 A가 실행 도중 네트워크로부터 데이터 도착을 기다리게 되었을 때, 즉 block 상태이므로 스레드 A는 네트워크로부터 데이터가 도착할 때까지 실행시키지 않는다. 대신 스레드 B나 C 중 하나를 선택하여 실행 기회를 준다.

6. ① sleep()

7. ② synchronized

8. (1) 스마트폰에서 문자를 전송하는 스레드와 음악을 연주하는 스레드의 경우, 두 스레드 사이에 데이터를 공유하지 않기 때문에 동기화가 필요하지 않다.
   (2) 미디어 플레이어에서, 하드디스크로부터 비디오 프레임을 읽어 공급하는 스레드와 읽은 프레임을 스크린에 그리는 스레드의 경우, 두 스레드는 비디오 프레

임 데이터를 공유하므로 동기화가 필요하다.

(3) 온도 센서로부터 주기적으로 값을 읽어 오는 스레드와 습도 센서로부터 주기적으로 값을 읽어오는 스레드의 경우, 온도 값과 습도 값은 서로 다른 데이터이므로 두 스레드 사이의 동기화가 필요없다.

(4) 웹 서버에서 접속한 사용자마다 로그인 정보를 데이터베이스에 기록하는 다수의 스레드의 경우, 데이터베이스에 접근하는 코드는 하나이므로 이 코드를 여러 스레드가 동시에 호출해야만 하므로 동기화가 필요하다.

9. ① 다른 스레드가 객체 obj의 notify()를 호출할 때

10.

```
class MyThread implements Runnable {
    public void run() { // 스레드 코드를 작성한다.
        try {
            Thread.sleep(5000); // 5초 동안 잠을 잔다.
        }
        catch(InterruptedException e) { return; }
    }
    public static void main(String [] args) {
        Thread th = new Thread(new MyThread());
        th.start(); // 스레드를 실행시킨다.
    }
}
```

14장 연습문제

이론문제

1. ④ Separator

2.

```
JMenuItem item1 = new JMenuItem("Open");
JMenuItem item2 = new JMenuItem("Save");
item1.addActionListener(new MyAction());
item2.addActionListener(new MyAction());

class MyAction implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        String s = e.getActionCommand();
        if(s.equals("Open")) ......; // 열기 작업을 수행한다.
        else ....; // 닫기 작업을 수행한다.
    }
}
```

3. ④. 툴바의 핸들을 마우스 드래깅할 수 없게 만드는 메소드는 JToolBar의 setFloatable(false)이다.

4. ③ .툴팁이 나타나지 않도록 설정하는 방법은 ToolTipManager의 setEnabled(false) 메소드를 사용한다.

5.

```
JButton b = new JButton("Hello");
b.setToolTipText("안녕하세요");
```

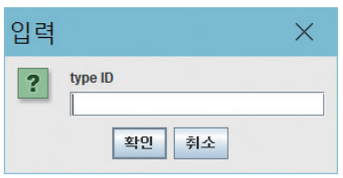
6.

```
JLabel la = new JLabel("Java");
la.setToolTipText("자바");
ToolTipManager m = ToolTipManager.sharedInstance();
m.setInitialDelay(1000);
m.setDismissDelay(10000);
```

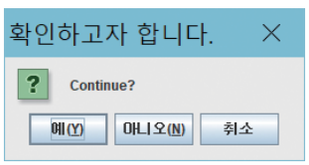
7. 모달 다이얼로그란 다이얼로그에 입력을 마치지 않고서는 다른 작업을 할 수 없도록 다이얼로그가 키 입력을 독점하는 것을 말한다. 일반적으로 파일을 처리하는 응용프로그램은 파일 열기 다이얼로그를 이용하여 사용자로부터 읽고자 하는 파일을 선택한다. 그러고 나서 응용프로그램은 사용자가 선택한 파일을 읽고 읽은 데이터를 가지고 계속 작업을 실행한다. 그러므로 파일 다이얼로그가 화면에 출력된 상태에서 사용자가 파일을 선택하지 않고 다른 작업을 시도하면 파일을 읽지 않았기 때문에 다른 작업을 하는 것이 무의미하다. 혹은 사용자가 파일 다이얼로그가 파일을 선택하는 과정을 생략한 채 다른 작업을 진행하도록 어설프게 응용프로그램을 작성하여서는 안 된다. 그러므로 파일 다이얼로그는 모달 다이얼로그로 작성되어야 한다.

8. ⑤ JTabbedPane

9. (1) JOptionPane.showInputDialog("type ID");



(2) JOptionPane.showConfirmDialog(null, "Continue?", "확인하고자 합니다.",
JOptionPane.YES_NO_CANCEL_OPTION);



10. ④. 탭팬에서 탭의 디폴트 위치는 위쪽, 즉 JTabbedPane.TOP이다.

15장 연습문제

이론문제

1. ipconfig 명령을 입력하면 현재 PC의 네트워크 설정 상황을 보여준다. ipconfig 명령을 입력하고 출력된 정보 중에서 "IPv4 주소" 줄에서 IP 주소를 확인할 수 있다.

2. ④. 개발자가 통신 프로그램을 작성할 때 잘 알려진 포트를 사용하면 기존의 포트와 충돌이 발생할 수 있으니 사용하지 않는 것이 바람직하다.

3. ④. 서버 소켓은 클라이언트로부터의 접속을 받기 위해 사용되는 소켓이며, 접속이 이루어지면 서버 소켓이 클라이언트와 통신을 할 수 있는 클라이언트 소켓을 생성해준다.

4. ②

5. ③. 5050은 접속하고자 하는 서버의 포트 번호이다.

6.

```
ServerSocket ss = new ServerSocket(5550); // 5550포트와 결합하는 서버 소켓 생성
Socket  s = ss.accept(); // 클라이언트로부터의 접속을 기다린다.
BufferedWriter out = new BufferedWriter(new
    OutputStreamWriter(s.getOutputStream()); // 소켓 s로 데이터를 전송할 출력 스트림
을 만든다.
out.write("안녕"); // 소켓으로 하여금 "안녕"을 전송하도록 한다.
s.close(); // 소켓 s를 닫는다.
ss.close(); // 서버 소켓을 닫는다.
```

7. ③. 생성된 ss 소켓은 클라이언트로부터의 연결을 받기 위해서만 사용된다.

8.

```
Socket s = new Socket("203.1.1.1", 6000); // 203.1.1.1주소의 6000번 포트에 접속을 시도한다.
BufferedReader in = new BufferedReader(new
    InputStreamReader(s.getInputStream()); // 소켓 s로부터 데이터를 수신할 입력 스트림을 만든다.
in.readLine(); // 소켓으로부터 한 라인의 텍스트("\n"으로 끝나는)를 입력받는다.
s.close(); // 소켓 s를 닫는다.
```

16장 연습문제

이론문제

1. ④. JDBC는 관계형 데이터베이스에 대한 API를 제공한다.

2. (1) 데이터베이스의 테이블은 행과 열로 구성된 2차원 배열 구조로서 데이터를 저장한다. 데이터베이스는 테이블들의 모임이다.
   (2) 테이블의 행은 연관된 여러 개의 속성으로 구성되며 레코드 또는 엔터티라고도 한다. 개념적 정보의 단위가 된다.
   (3) 테이블의 열은 속성을 나타내며 속성들이 모여 엔터티의 성질이나 상태를 나타낸다.

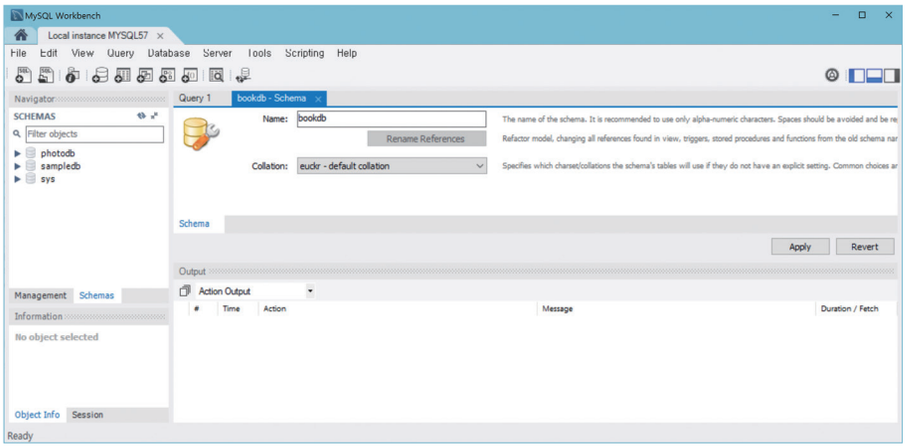
3. SQL

4. (1) Statement 클래스는 SQL 문을 실행하기 위해 사용하는 클래스이다. 데이터를 검색하기 위한 executeQuery() 메소드를 제공하고 데이터 변경을 위한 executeUpdate() 메소드를 제공한다.
   (2) ResultSet 클래스는 SQL 문 실행 결과를 얻어오기 위해 사용하는 클래스이다. ResultSet 객체는 현재 레코드 위치를 가리키는 커서를 관리하며, 커서의 위치와 관련된 메소드와 레코드를 가져오는 메소드를 제공한다.

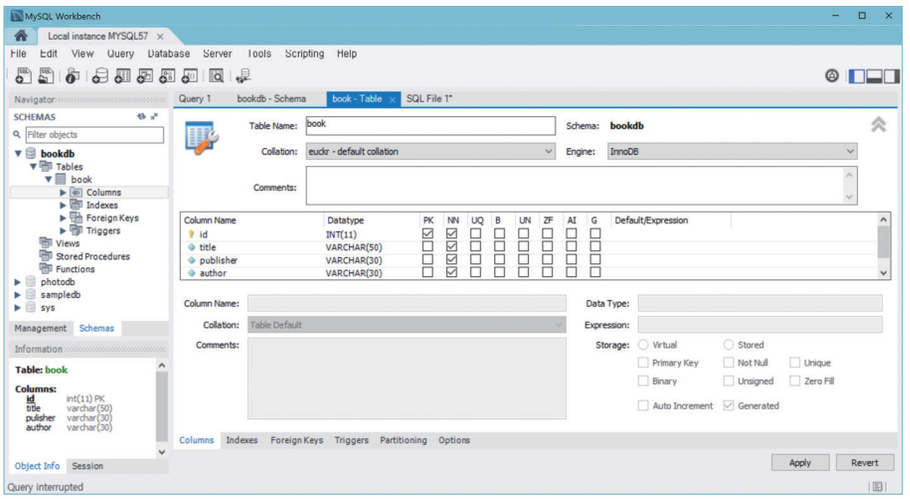
5. 열의 이름을 인자로 하거나 또는 열의 인덱스를 인자로 하여 각각 getInt("id") 또는 getInt(1)를 빈칸에 삽입할 수 있다.

실습문제

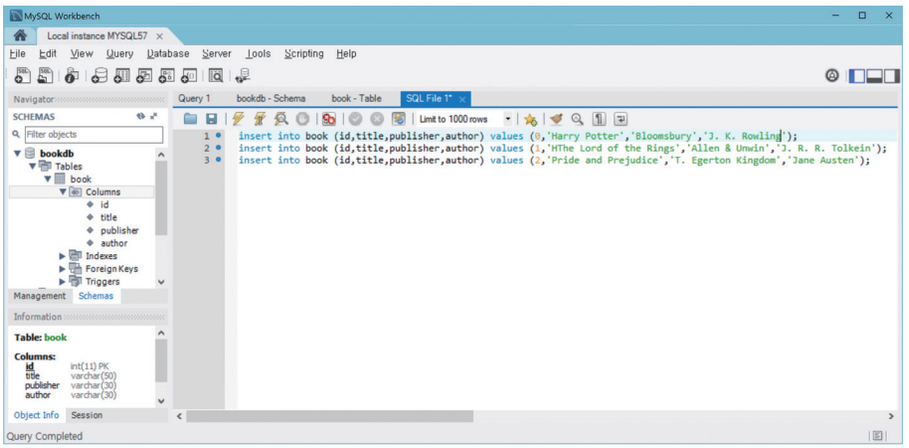
1.


MySQL Workbench
Local instance MYSQL57
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
photodb
sampledb
sys
Query 1
bookdb - Schema
Name: bookdb
Rename References
Collation: euckr - default collation
Schema
Apply
Revert
Output
Action Output
Time Action Message Duration / Fetch
Management Schemas
Information
No object selected
Object Info Session
Ready

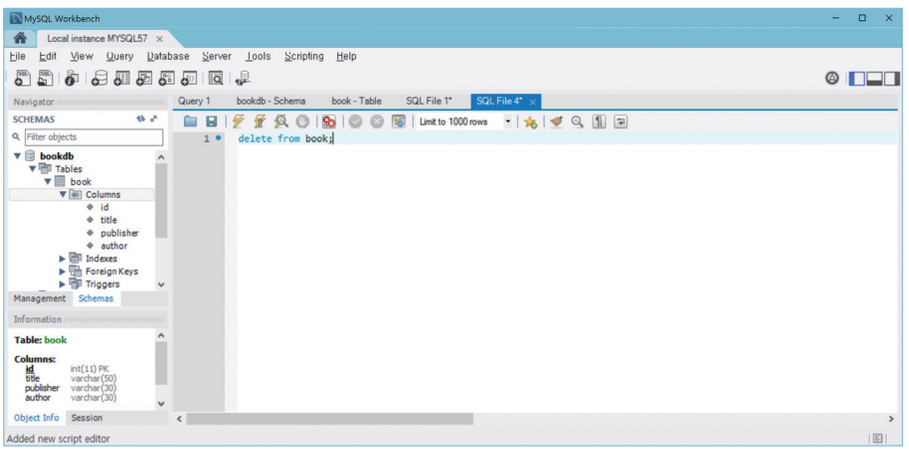
2.


MySQL Workbench
Local instance MYSQL57
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
bookdb
Tables
book
Columns
Indexes
Foreign Keys
Triggers
Views
Stored Procedures
Functions
photodb
sampledb
sys
Query 1
bookdb - Schema
book - Table
SQL File 1*
Table Name: book
Schema: bookdb
Collation: euckr - default collation
Engine: InnoDB
Comments:
Column Name Datatype PK NN UQ B UN ZF AI G Default/Expression
id INT(11)
title VARCHAR(50)
publisher VARCHAR(30)
author VARCHAR(30)
Column Name:
Collation: Table Default
Comments:
Data Type:
Expression:
Storage: Virtual Stored
Primary Key Not Null Unique
Binary Unsigned Zero Fill
Auto Increment Generated
Columns Indexes Foreign Keys Triggers Partitioning Options
Apply
Revert
Management Schemas
Information
Table: book
Columns:
id int(11) PK
title varchar(50)
pulisher varchar(30)
author varchar(30)
Object Info Session
Query interrupted

3.


MySQL Workbench
Local instance MYSQL57
File Edit View Query Database Server Tools Scripting Help
Query 1 bookdb - Schema book - Table SQL File 1*
Limit to 1000 rows
1 insert into book (id,title,publisher,author) values (0,'Harry Potter','Bloomsbury','J. K. Rowling');
2 insert into book (id,title,publisher,author) values (1,'HThe Lord of the Rings','Allen & Unwin','J. R. R. Tolkein');
3 insert into book (id,title,publisher,author) values (2,'Pride and Prejudice','T. Egerton Kingdom','Jane Austen');
SCHEMAS
Filter objects
bookdb
Tables
book
Columns
id
title
publisher
author
Indexes
Foreign Keys
Triggers
Management Schemas
Information
Table: book
Columns:
id int(11) PK
title varchar(50)
publisher varchar(30)
author varchar(30)
Object Info Session
Query Completed

4.


MySQL Workbench
Local instance MYSQL57
File Edit View Query Database Server Tools Scripting Help
Query 1 bookdb - Schema book - Table SQL File 1* SQL File 4*
Limit to 1000 rows
1 delete from book;
SCHEMAS
Filter objects
bookdb
Tables
book
Columns
id
title
publisher
author
Indexes
Foreign Keys
Triggers
Management Schemas
Information
Table: book
Columns:
id int(11) PK
title varchar(50)
publisher varchar(30)
author varchar(30)
Object Info Session
Added new script editor