연습문제 이론 정답

1장 연습문제

이론 문제

1. 자바 소스의 확장자는 .java이고 컴파일된 클래스 파일의 확장자는 .class이다.

2. 자바 이전의 언어들은 플랫폼 종속적이어서, 한 플랫폼에서 컴파일한 목적 코드는 오직 그 플랫폼에서만 실행되기 때문에, 하나의 소스 프로그램을 플랫폼마다 컴파일하는 번거로움이 있었다. 자바는 한 번 작성하면 플랫폼에 관계없이 바로 실행할 수 있는 플랫폼 독립적인 언어와 실행 환경을 구성하기 위해 개발되었다. 특별히, 다양한 하드웨어 및 운영체제 플랫폼을 가지고 있고, 적은 양의 메모리가 탑재되는 가전제품에 대해, 한 번 작성하면 플랫폼에 상관없이 실행될 수 있는 새로운 언어와 실행 체계를 위해 개발된 언어이다.

3. WORA(Write Once Run Anywhere)

4. ④

5. JRE는 자바 프로그램의 실행 환경으로 자바 가상 기계를 포함하고 있으며 자바 실행 환경만 필요한 경우에 사용되며, JDK는 자바 컴파일러, 도구, 라이브러리 등 자바 프로그램 개발에 필요한 모든 것과 JRE를 포함한다. 따라서 자바 응용프로그램 개발을 위해서는 JDK가 필요하다.

6. ④

7. ③

8. ②

9. (1) W.java에 저장되어야 한다.
   (2) 총 4개의 클래스 파일(W.class, W$X.class, Y.class, Z.class)이 생성된다.

10. 자바 프로그램이 저장되는 소스 파일의 이름은 Calc.java이고, 이 소스가 컴파일되면 Calc.class 파일이 생성된다.

2장 연습문제

이론 문제

1. 자바에서 클래스를 선언할 때는 class 키워드를 사용한다.

2. 잘못된 식별자와 이유는 다음과 같다.

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

3. 올바른 변수 선언은 다음과 같다.
   (1) int age;
   (2) float f = 0.25F;
   (3) double d = age + f; 또는 double d = (double)age + (double)f
   (4) char c = 'a';
   (5) String name = "황기태";

4. 수식의 결과 값과 타입은 다음과 같다.
   (1) 67 + 12.8 -> double 타입의 79.8
   (2) 'c' + 1 -> int 타입의 100
   (3) 10/3 -> int 타입의 3
   (4) 10.0/3 -> double 타입의 3.333333333333333
   (5) 10==9 -> boolean 타입의 false

5. (1) a는 b보다 크거나 같다. -> a >= b 또는 a > b || a == b
   (2) a는 b보다 작고 c보다 크다. -> a < b && a > c
   (3) a 더하기 3은 10과 같지 않다. -> (a + 3) != 10
   (4) a는 10보다 크거나 b와 같다. -> (a > 10) || (a == b)

6. ④ 5.4
   현재 JDK8 기준으로 case 문의 값으로 사용할 수 있는 리터럴은 정수, 문자, 문자열뿐이다. 그러므로 실수 리터럴로 사용할 수 없다.

7. (1) SampleProgram.java

(2) SampleProgram 클래스에 main() 메소드가 없기 때문에 오류가 난다. main() 을 삽입하여 다음과 같이 작성하면 된다.

```
public class SampleProgram {
   public static void main(String[] args) {
      int i;
      int j;
      i = 20;
      j = 30;
      System.out.println(i+j);
   }
}
```

8. (1) case 1로 분기하여 break를 만날 때까지 실행되므로 다음과 같이 출력됨
옵션 1
옵션 2
옵션 3
(2) case 2로 분기하여 break를 만날 때까지 실행되므로 다음과 같이 출력됨
옵션 2
옵션 3
(3) case 3으로 분기하여 break를 만날 때까지 실행되므로 다음과 같이 출력됨
옵션 3
(4) default 문으로 분기하므로 다음과 같이 출력됨
해당 없음

9. 다음과 같이 한 줄로 작성할 수 있다.

```
i = (j%2 == 0)?10:20;
```

10. if-else 문을 switch 문으로 바꾸면 다음과 같다.

```
switch(i) {
   case 1 : System.out.println("!"); break;
   case 2 : System.out.println("@"); break;
   case 3 : System.out.println("#"); break;
   default: System.out.println("*");
}
```

3장 연습문제

이론 문제

1. (1) 0에서 10까지(혹은 2에서 10까지) 짝수만 더하는 프로그램이며, 실행 결과는 30
(2)

```
int i=0, sum=0;
while(true) {
   i = i + 2;
   sum += i;
   if(i == 10) break;
}
System.out.println(sum);
```

```
int i=0, sum=0;
do {
   i = i + 1;
   if(i%2 = 1) continue;
   sum += i;
} while(i<10);
System.out.println(sum);
```

2. (1) 12.2
(2) for(int i=0; i<d.length; i++)
(3) while 문으로 바꾸어 작성하면 다음과 같다.

```
double sum = 0.0;
double d[] = {1.0, 2.3, 3.4, 5.5 };
int i=0;
while(i<d.length) {
   sum += d[i];
   i++;
}
System.out.println(sum);
```

(4) do-while 문으로 바꾸어 작성하면 다음과 같다.

```
double sum = 0.0;
double d[] = {1.0, 2.3, 3.4, 5.5 };
int i=0;
do {
   sum += d[i];
   i++;
} while(i<d.length);
System.out.println(sum);
```

⑸ for-each 문으로 바꾸어 작성하면 다음과 같다.

```
double sum = 0.0;
double d[] = {1.0, 2.3, 3.4, 5.5 };
for(double x : d) {
   sum += x;
}
System.out.println(sum);
```

3. 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'};
⑷ double [][] d = new double[5][4];
⑸ int val [][] = {{1,2,3,4}, {5,6,7,8}, {9,10,11,12}};

4. myArray의 인덱스 범위는 0 ~ myArray.length-1이므로 다음 보기는 오류이다.
② myArray[myArray.length] = 100;

5. 다음 2차원 배열 선언문에서 문법적으로 잘못된 것은 다음 보기이다.
④ int [3][2] n = { {1,2}, {3,4}, {4,5} }; // int [3][2] n에서 3,2를 사용하면 안 됨

6.

```
boolean [] b = { true, false, true, true };
for( boolean boo : b )
   System.out.println(boo);
```

7. ⑴ 빈칸을 채우면 다음과 같다.

```
double [] allocArray() {
   Scanner scanner = new Scanner(System.in);
   double [] n = new double [scanner.nextInt()]; // 입력된 정수 크기의 배열 생성
   return n; // 배열 리턴
}
```

⑵ double d [] = allocArray();

8. (1) 코드를 실행한 결과 출력되는 내용은 다음과 같다.

```
124
계산을 끝냅니다.
```

(2)

```
23.5를 정수로 변환할 수 없습니다.
계산을 끝냅니다.
```

4장 연습문제

이론 문제

1. ④
   클래스의 멤버 변수들은 보호하기 위해 가능하면 private으로 선언하는 것이 바람직하다.

2. ③
   this는 생성자에서 사용 가능하다.

3. ③

4. 빈칸에 적절한 말을 삽입하면 다음과 같다.

   > static 멤버는 클래스 멤버라고도 불리며, 동일한 클래스의 모든 객체들의 의해 공유 된다. static 멤버는 객체 레퍼런스나 클래스 이름으로도 접근할 수 있으며, 객체가 생성되기 전 프로그램이 시작될 때 생성되어 있다. 특히 static 메소드는 static 멤버들만 접근할 수 있고, this 을(를) 사용할 수 없는 제약 사항이 있다.

5. 두 메소드 f()의 매개 변수 개수를 다르기 때문에, f()의 메소드 오버로딩은 성공한 경우이다. 리턴 타입이 같거나 다르건 오버로딩에 상관없다.

6. 빈칸에 적절한 말을 삽입하면 다음과 같다.

   > 자바에서는 객체를 임의로 소멸시킬 수 없으며, 이것은 개발자에게 매우 다행한 일이다. 참조하는 레퍼런스 가 하나도 없는 객체를 가비지라고 판단하고, 가비지를 가용 메모리로 자동 수집하는 가비지컬렉션 을 진행시킨다. 응용 프로그램에서 자바 플랫폼에 이 과정을 지시하고자 하면 System.gc() 코드를 호출하면 된다.

7. 가능하면 멤버 변수(필드)는 private으로 선언하고, 생성자나 메소드를 통해 접근하도록 하는 것이 바람직하다. 생성자를 이용하면 다음과 같이 할 수 있다.

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

또는 다음과 같이 메소드를 작성해도 된다.

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

8. 프로그램의 실행 결과는 다음과 같이 출력된다.

```
15
```

plusTen() 메소드의 매개 변수 x는 원본 ob 객체를 가리키므로, x로 ob 객체의 멤버 n 값을 마음대로 바꿀 수 있다. plusTen() 메소드에서 객체 ob의 n 값을 증가시켜 15가 된다.

9.

```
public class Rectangle {
   int w, h;
   Rectangle(int w, int h) {
      this.w = w; this.h = h;
   }
    Rectangle(int w) {
      this.w = w; this.h = 2;
   }
    Rectangle() {
      this.w = 1; this.h = 2;
   }
}
```

10. ④

read() 메소드가 리턴하면 변수 s가 사라지며, 따라서 new Scanner(System.in); 에 의해 생성된 객체는 자기를 가리키는 레퍼런스가 하나도 없어지기 때문에 가비지가 된다.

11. ④

getB()는 non-static 메소드이므로, static 메소드인 g()에서 호출할 수 없다.

12. ①

StaticSample 클래스의 멤버 x는 non-static 타입이므로 클래스 이름으로 접근할 수 없다.

5장 연습문제

이론 문제

1. ⑴ objA의 멤버들은 int a, int b, void set()
⑵ objB의 멤버들은 int a, int b, void set(), int c, int d
⑶ objC의 멤버들은 int a, int b, void set(), int c, int d, int e, int f

2. ②
슈퍼 클래스의 protected 멤버는 같은 패키지에 있든 아니든 서브 클래스는 접근 가능하며, 서브 클래스가 아니더라도 같은 패키지에 있는 클래스는 접근 가능하다.

3. ④
오버라이딩된 메소드가 호출되면 동적 바인딩이 발생한다.

4. ③
서브 클래스에서 슈퍼 클래스의 private 멤버를 직접 접근할 수 없다.

5.

```
class LCD {
   private int size;
   public LCD(int n) { size = n; }
}
class ColorLCD extends LCD {
   int colorSize;
   public B(int colorSize, int size) {
      super(size);
      this.colorSize = colorSize;
   }
}
```

6. (1) 컴파일 오류가 발생하는 곳은 아래 코드에 표기되어 있다.

```
class A {
   public A(int x) { System.out.print("A")+ x; }
}
class B extends A {
   public B() { System.out.print("B"); } // 컴파일 오류 발생
   public B(int x) { System.out.print("B" + x); } // 컴파일 오류 발생
}
```

컴파일 오류가 발생하는 이유는 컴파일러가 생성자 B()와 B(int x)를 컴파일할 때, 클래스 A의 디폴트 생성자 A()를 호출하도록 컴파일하지만, 클래스 A에는 디폴트 생성자 A()가 선언되어있지 않기 때문이다.

(2) 생성자 B()를 다음과 같이 수정한다.

```
public B() {
   super(20);
   System.out.print("B");
}
```

(3) 생성자 B(int x)를 다음과 같이 수정한다.

```
public B(int x) {
   super(x+20); // 혹은 super(50);
   System.out.print("B" + x);
}
```

7.

(1) ②

(2) new A()에 의해 생성된 객체는 B를 상속받지 않았기 때문에 (B)new A();와 같이 B 클래스의 객체로 다운 캐스팅될 수 없기 때문이다.

8. (1) 다음과 같이 두 라인이 출력된다.

```
false
true
```

(2) 다음과 같이 두 라인이 출력된다.

```
false
true
```

9. 빈칸에 들어가는 코드는 다음과 같다.
(1) `draw();`
(2) `super.draw();`

10. ①

11. (1) ②, ④
(2)

```
class Circle extends Shape {
   private int radius;

   public Circle(int radius) { this.radius = radius; }
   double getArea() { return 3.14*radius*radius; }
   public void draw() { System.out.println("반지름="+radius); }
}
```

12. ③
인터페이스에는 필드를 선언할 수 없다. 인터페이스는 추상 메소드와 상수로만 구성된다.

6장 연습문제

이론 문제

1. java.lang 패키지에 속한 클래스들은 import 문 없이 사용할 수 있다.

2. Scanner와 StringTokerizer 클래스의 이름을 패키지명을 포함하는 완전 경로명으로 사용해야 한다. 다음과 같다.

```
public class Example {
   public static void main(String[] args) {
      java.util.Scanner scanner = new java.util.Scanner(System.in);
      java.util.StringTokenizer st = new
java.util.StringTokenizer(scanner.nextLine(), "/");
      while(st.hasMoreTokens())
         System.out.println(st.nextToken());
   }
}
```

3. Circle 클래스를 drawable 패키지에 속하게 하고자 하기 위해, 다음과 같이 package 문을 첫 줄에 삽입한다.

```
package drawable;
public class Circle {
   int radius;
   public Circle(int radius) { this.radius = radius; }
}
```

그리고 Main 클래스를 app 패키지에 저장하고, Circle 클래스를 사용하기 위해 다음 코드와 같이 한다.

```
package app;
import drawable.Circle;
public class Main {
   public static void main(String[] args) {
      Circle c = new Circle(5);
   }
}
```

4. (1) `int i = Integer.parseInt("20");`
(2) `double d = Double.parseDouble("35.9");`
(3) `String s = Boolean.toString(true);`
(4) `if(Character.isAlphabetic(c)) System.out.println("eng");`
(5) `String s = Integer.toBinaryString(n);`
(6) `String s = Integer.toHexString(50);`

5. (1) `Integer n = new Integer(20);`
`new Integer(20);` 부분은 정수 20을 박싱하는 코드이다.
(2) `double d = 1.2 + new Double(3.4);`
`new Double(3.4);`은 실수 3.4를 박싱하는 코드이며, 1.2와의 덧셈을 위해 `new Double(3.4)`의 객체는 자동으로 언박싱되어 3.4가 된다. 그리고 1.2와 3.4가 더해져서 4.6이 된다.
(3) `System.out.print(3 + new Integer(20));`
`new Integer(20)`는 정수 20을 박싱한 코드이며, 3과 더하기를 위해 다시 자동 언박싱되어 20으로 처리된다. 3과 20이 더해져서 23이 되어 `System.out.print(23)`에 의해 출력된다.
(4) `Boolean b = true;`
`true`가 `new Boolean(true)`로 자동 박싱된다.
(5) `float f = new Float("10.1");`
`new Float("10.1")`으로 실수 10.1이 박싱되고, `float` 타입의 변수 `f`에 치환되기 위해 `new Float("10.1")`은 10.1로 자동 언박싱된다.
(6) `String s = "abc";`
`String` 클래스는 `Wrapper` 클래스가 아니므로 박싱 혹은 언박싱과 무관하다.

6. 두 레퍼런스가 같은지 비교하기 위해 사용되는 것은 == 연산자이며, 두 레퍼런스가 가리키는 객체의 내용물이 같은지 비교하기 위해 사용되는 것은 `equals()` 메소드이다.

7. (1) a와 == 연산을 수행하였을 때 `true`가 되는 문자열을 b이다. `"Hello"`는 리터럴 테이블에 저장되기 때문에 a와 b 레퍼런스의 값은 같다.
(2) f와 `equals()` 연산을 수행하였을 때 `true`가 되는 문자열은 c, e이다. `equals()`는 객체의 내용을 비교하므로 c와 e는 f와 같은 문자열이다.

8.

```
StringTokenizer st = new StringTokenizer("2+3+4+6", "+");
int n = st.countTokens();
System.out.println("토큰 개수 = " + n);
while(st.hasMoreTokens()) { // st에 남은 토큰이 있는 동안
   System.out.print( st.nextToken() + " "); // st로부터 토큰 얻어내고 출력
}
```

9. 코드 각 라인이 실행될 때 a, b, c가 변하는 것을 주석에 설명하였다.

```
String a = new String("  hello  "); // a = "  hello  "
String b = a; // a = "  hello  ", b = "  hello  "
String c = a.trim(); // c = "hello". a = "  hello  "
a = a.toUpperCase(); // a = "  HELLO  "
```

최종적으로 a, b, c는 다음과 같다.

```
a = "  HELLO  "
b = "  hello  "
c = "hello"
```

7장 연습문제

이론 문제

1. ④
   컬렉션은 배열과 달리 요소의 개수가 가변적이다.

2. ②
   제네릭은 C++, C# 등 현존하는 많은 언어에서 지원을 확대하고 있으며, 제네릭 프로그래밍은 점점 확산되고 있다.

3. ①
   v.size()는 현재 삽입되어 있는 요소의 개수를 리턴하므로 현재 삽입된 요소가 없으면 0을 리턴한다. new Vector<Integer>(3)에 주어진 3은 초기 벡터의 용량을 지정하는 것으로, 벡터의 현재 용량을 알고자 하면 v.capacity()를 호출하면 된다.

4. ⑴ HashMap<String, Double> h = new HashMap<String, Double>();
   ⑵ Vector<String> v = new Vector<String>();
   ⑶ ArrayList<Circle> a = new ArrayList<Circle>();

5. 제네릭 컬렉션을 사용할 때 제네릭 타입에 대입할 수 있는 타입으로 int, char 등의 기본 타입은 사용할 수 없다. 그러므로 다음과 같이 수정해야 한다.

```
Vector<Integer> v = new Vector<Integer>(100);
```

6.

```
Vector<Integer> v = new Vector<Integer>();
Iterator<Integer> it = v.iterator();
```

7.

```
ArrayList<Double> a = new ArrayList<Double>();
a.add(3.5); // 3.5가 new Double(3.5)로 자동 박싱된다.
double d = a.get(0); // a.get(0)가 리턴하는 Double 객체를 double 타입으로 만들기 위해
                        a.get(0)을 a.get(0).doubleValue()로 자동 언박싱된다.
```

8.

```
ArrayList<Double> a = new ArrayList<Double>();
for(int i=0; i<20; i++) {
   double d = Math.random()*100; // 0.0 ~ 99.999 사이의 랜덤 실수
   a.add(d);
}

// Iterator를 이용하여 수정한 정답은 다음과 같다.
Iterator<Double> it = a.iterator();
while(it.hasNext())
   System.out.println(it.next());
```

9. 빈칸에 코드를 채우면 다음과 같다.

```
Vector<String> v = new Vector<String>();
v.add("Good"); // v에 "Good" 삽입
v.add("Bad"); // v에 "Bad" 삽입
System.out.println(v.size()); // v에 현재 삽입된 문자열 개수 출력
v.remove(1); // v의 인덱스 1에 있는 "Bad" 문자열 삭제
```

10. 빈칸에 코드를 채우면 다음과 같다.

```
// 사람 이름의 키로 하고 나이를 값으로 다루는 해시맵 생성
HashMap<String, Integer> h = new HashMap<String, Integer>(); // 해시맵 생성
h.put("이몽룡", 25); // h에 "이몽룡" 나이 25 저장
h.put("성춘향", 18); // h에 "성춘향" 나이 18 저장
Scanner scanner = new Scanner(System.in);
String name = scanner.next(); // 사용자로부터 이름 입력
Integer age = h.get(name); // h에서 name의 나이 검색
```

11. 4개의 문항에 대해 답하면 다음과 같다.

```
class MyGeneric<W> {
   private W x;
   public MyGeneric(W x) {
      this.x = x;
   }
   public W take() { return x; } // (2)
   public boolean compare(W x) { // (3)
      if(this.x.equals(x)) return true;
      else return false;
```

```
    }
}
```

⑴ MyGeneric의 타입 매개 변수는 W이다.
⑵ take() 메소드 코드 정답은 앞의 소스에 있다.
⑶ compare() 메소드 코드 정답은 앞의 소스에 있다.
⑷ String으로 구체화한 MyGeneric 객체를 생성하고 활용 예를 들면 다음과 같다.

```
MyGeneric<String> g = new MyGeneric<String>("Kitae");
System.out.println(g.take()); // "Kitae"를 출력한다.
System.out.println(g.compare("Kito")); // false를 출력한다.
```

8장 연습문제

이론 문제

1. ① Panel

2. ③ Font

3. 코드의 빈칸을 채우면 다음과 같다.

```
import java.awt.*;
import javax.swing.*;
public class MyFrame extends JFrame {
   MyFrame() {
      Container c = getContentPane(); // 컨텐트팬에 대한 레퍼런스 얻기
      c.add(new JButton("hello")); // 컨텐트팬에 "hello" 버튼 달기
      setSize(200, 400);  // 프레임을 너비 200, 높이 400픽셀로 설정
      setVisible(true);
   }
   public static void main(String [] args) {
      MyFrame frame = new MyFrame(); // MyFrame 생성
   }
}
```

4.

```
import javax.swing.*;
public class MyFrame extends JFrame {
   MyFrame() {
      setSize(300,300);
      setVisible(true);
   }
   public static void main(String [] args) {
      // JFrame mf = new JFrame(); // 이 문장을 아래와 같이 수정한다.
      MyFrame frame = new MyFrame();
   }
}
```

5. ②
컨테이너는 다른 컨테이너에 삽입될 수 있다.

6. ②
배치관리자는 한 컨테이너에 최대 한 개만 존재한다.

7. (1)

```
Container c = getContentPane(); // 컨텐트팬 알아내기
c.setLayout(new BorderLayout(10, 20)); // 배치관리자 설정
```

(2)

```
Container c = getContentPane(); // 컨텐트팬 알아내기
c.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 20)); // 배치관리자 설정
```

(3)

```
Container c = getContentPane(); // 컨텐트팬 알아내기
c.setLayout(new GridLayout(2, 5, 10, 20)); // 배치관리자 설정
```

8.

```
import java.awt.*;
import javax.swing.*;
public class MyFrame extends JFrame {
   MyFrame() {
      Container c = getContentPane(); // 컨텐트팬에 대한 레퍼런스 얻기
      c.setLayout(null); // 컨텐트팬의 배치관리자 제거
      JButton b = new JButton("Hello");
      b.setLocation(30, 40); // b의 위치를 30,40으로 설정
      b.setSize(100, 100); // b의 크기를 100x100으로 설정
      c.add(b); // 컨텐트팬에 b 삽입
      setSize(300,300);
      setVisible(true);
   }
   public static void main(String [] args) {
      new MyFrame();
   }
}
```

9장 연습문제

이론 문제

1. ④
   이벤트 리스너는 익명 클래스나 내부 클래스 혹은 외부 클래스로 작성할 수 있으며, 외부 클래스의 경우 별도의 자바 파일에 작성할 수 있다. 그러나 반드시 별도의 자바 파일에 작성할 필요는 없다.

2. ②
   마우스 드래깅 길이는 MouseEvent 객체에 저장되지 않는 정보이다.

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. 틀린 부분을 수정하면 다음과 같다.

```
class MyActionListener implements ActionListener {
   public void actionPerformed(ActionEvent e) {
      System.out.println("Click");
   }
}
```

6. 틀린 부분을 수정하면 다음과 같다.

(1)

```
class MyMouseListener extends MouseAdapter {
   public void mousePressed(MouseEvent e) {
      System.out.println("Mouse Pressed");
   }
}
```

(2)

```
class MyKeyListener extends KeyAdapter {
   public void keyTyped(KeyEvent e) {
      System.out.println("Key Typed");
   }
}
```

7. 유니코드 키가 아닌 것은 다음과 같다.

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

8. <Esc> 키는 유니코드 키가 아니므로 keyTyped()는 호출되지 않는다. 그러므로 호출되는 순서는 keyPressed(), keyReleased() 순이다.

9.

```
la.addMouseListener(new MyMouseListener()); // la에 마우스 리스너를 등록한다.
...
class MyMouseListener extends MouseAdapter { // 마우스 리스너를 선언한다.
   public void mouseReleased(MouseEvent e) { // 눌러진 마우스가 놓여지는 순간 처리
      JLabel label = (JLabel)e.getSource(); // 이벤트 소스를 알아낸다.
      label.setText("안녕");          // 문자열을 "안녕"으로 변경한다.
   }
}
```

* mouseReleased 대신 mouseClicked를 써도 된다.

10. 빈칸을 채우면 다음과 같다.

```
public void keyPressed(KeyEvent e) {
   if(e.getKeyCode() == KeyEvent.VK_DELETE) // Delete 키가 눌러진 경우
      System.out.println("Delete");
   else if(e.getKeyChar() == '#') // # 키가 눌러진 경우
      System.out.println("#");
}
```

11. ③ c.requestFocus();

10장 연습문제

이론 문제

1. ②

   Container 클래스는 AWT 패키지에 속하는 클래스이다.

2. ④

   JTextField에 <Enter> 키를 입력하면 ActionEvent가 발생한다.

3. ③

   JCheckBox 컴포넌트를 마우스로 선택하면 Item 이벤트가 발생한다.

4. 빈칸에 적절한 코드를 삽입하면 다음과 같다.

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

5. 메뉴를 만들어 프레임에 붙이는 코드의 빈칸을 채우면 다음과 같다.

```
JMenuBar mb = new JMenuBar();
JMenu fileMenu = new JMenu("File"); // "File" 메뉴를 생성한다.
mb.add(fileMenu); // 메뉴바에 파일 메뉴를 붙인다.
fileMenu.add(new JMenu("New")); // "New" 메뉴아이템을 생성하여 붙인다.
fileMenu.addSeparator(); // 분리선을 삽입한다.
frame().setJMenuBar(mb); // 프레임에 메뉴바를 붙인다.
```

6.

간단한 팝업 다이얼로그를 생성하여 화면에 출력하려면 JOptionPane 클래스를 이용하면 된다. 이 클래스에는 팝업 다이얼로그를 출력하는 static 타입의 메소드가 여러 개 있다. 하지만 이들은 모두 모달 다이얼로그로서 일

단 화면에 출력되면 닫기 전에 다른 작업을 할 수 없는 특징이 있다. 이름이나 주소 등 한 줄로 된 문자열을 입력받을 수 있는 간단한 입력 다이얼로그는 showInputDialog() 메소드를 호출하면 된다. 이 메소드의 리턴 값은 사용자입력 문자열이지만, 취소 버튼이 선택되거나 강제로 창이 닫히면 null를(을) 리턴한다.

7. "sunny.jpg"를 가진 이미지 레이블 sunnyLabel을 만드는 코드는 다음과 같다.

```
ImageIcon icon = new ImageIcon("sunny.jpg"); // 이미지 파일 로딩, 이미지 객체 생성
JLabel sunnyLabel = new JLabel(icon); // 레이블 컴포넌트 sunnyLabel 생성
```

8.

```
JTextArea ta = new JTextArea();
Container c = getContentPane(); // 프레임의 컨텐트팬을 알아낸다.
JScrollPane scrollPane = new JScrollPane(ta); // JScrollPane에 ta를 붙인다.
c.add(scrollPane); // scrollPane을 컨텐트팬에 붙인다.
```

9. (1) JRadioButton, 3개
'아침', '점심', '저녁'을 나타내는 3개의 라디오 버튼을 사용하여 하나를 선택하도록 한다.
(2) JCheckBox, 3개
방법 1) '남/여'를 위해 1개, '내국인/외국인' 선택을 위해 1개, '성년/미성년' 선택을 위해 1개 총 3개의 체크박스를 둔다.
방법 2) 물론 JRadioButton을 6개 사용하여, 두 개는 '남/여' 선택, 두 개는 '내국인/외국인' 선택, 두 개는 '성년/미성년' 선택에 사용하게 할 수도 있지만, 총 6개의 라디오 버튼을 사용하고 각 두 개씩 버튼 그룹으로 묶어야 하므로 방법 1보다는 비효율적이다.
결론적으로 방법 1의 JCheckBox, 3개가 정답이다.
(3) JLabel, 4개
이미지는 출력하기에 적절한 컴포넌트는 JLabel이고 이미지가 총 4개이기 때문이다.
(4) JButton, 1개
'다음'으로 계속 진행을 지시하기 위해서는 JButton 컴포넌트 1개가 필요하며, Action 이벤트 리스너를 달면 된다.

11장 연습문제

이론 문제

1. ③ paintComponent(Graphics g)
   public void paintComponent(Graphics g)는 JComponent의 메소드이다.

2. ① JPanel

3. ④ 애니메이션

4. ③
   자바에서 Graphics로 선을 그릴 때 선의 두께를 마음대로 조절할 수 없다.

5. (1) g.drawImage(img, 10, 20, null);
   (2) g.drawImage(img, 10, 10, getWidth()-20, getHeight()-20, null);

6. (1)
```
g.setFont(new Font("Times New Roman", Font.PLAIN, 30));
g.drawString("We Win!!", 100, 100);
```

   (2)
```
g.setColor(Color.BLUE);
g.drawOval(0, 0, getWidth()-1, getHeight()-1);
```

7. 컴포넌트를 이용하여 GUI를 구성하면 컴포넌트 모양의 한계를 벗어날 수 없어 정형화된 모양의 GUI 밖에 구성할 수 없다. 하지만 그래픽을 이용하면 컴포넌트로 표현할 수 없는 모양과 방식으로 GUI를 구성할 수 있다. 다만 개발자가 일일이 그래픽으로 화면을 구성해야 하는 어려움이 있다.

8. 컴포넌트의 repaint() 메소드를 호출하면 자바 플랫폼은 컴포넌트의 paintComponent()를 호출한다.

9. super.paintComponent()는 Jpanel의 paintComponent()를 호출하는 코드이다. Jpanel의 paintComponent()는 라인 7에서 배경색으로 지정된 노란색으로 패널의 배경을 칠해 이전에 그려진 내용을 모두 지운다. 만일 super.paintComponent()를 호출하지 않는다면, 이전에 그려진 잔상이 지워지지 않고 배경색도 칠해지지 않게 된다.

12장 연습문제

이론 문제

1. 영화 보면서 팝콘 먹기, 전화하면서 문서 작성하기

2. ②
자바에서는 멀티스레드만 지원한다.

3. (1) `public void run();`
(2) `public void start();`
(3) `public void interrupt();`

4. 10초 후에 종료하는 프로그램 코드이다.

```
class MyThread implements Runnable {
   public void run() { // 스레드 코드를 작성한다.
      try {
         Thread.sleep(10000); // 10초 동안 잠을 잔다.
      } catch(InterruptedException e) { return; }
   }
   public static void main(String [] args) {
      Thread th = new Thread(new MyThread());
      th.start(); // 스레드를 실행시킨다.
   }
}
```

5. JVM이 자바 응용프로그램을 실행하기 위해 생성하는 스레드는 main 스레드(메인 스레드)이다. 메인 스레드는 자바 응용프로그램의 main() 메소드에서부터 실행을 시작한다. main() 메소드의 실행을 끝내면 메인 스레드는 스스로 종료한다.

6. ③ Object
wait(), notify()는 java.lang.Object 클래스의 메소드이다.

7. ②
스마트폰에서 문자를 전송하는 스레드와 음악을 연주하는 스레드는 서로 공유하는 데이터가 없기 때문에 동기화 될 필요가 없다.

8. ③

웹 서버에는 접속하는 각 사용자에 대해 스레드가 서비스하도록 할 필요가 있다. 또한 그래픽 편집기에서 프린팅 하는 동안 그리기 이루어지려면 프린팅을 맡은 스레드와 그리기 스레드가 따로 작동될 필요가 있다.

하지만 1에서 100000까지 더하기를 할 때, 짝수만 더하는 스레드와 홀수만 더하는 스레드를 두는 경우를 보자. 컴퓨터가 두 개의 스레드를 동시에 처리한다고 하더라도 하나의 CPU만 가진 경우 시간을 분할하여 두 개의 스레드를 돌아가면서 실행할 수밖에 없다. 그러므로 실행하고자 하는 스레드가 입출력 작업이 많아 대기하는 시간이 많은 경우, 대기하는 시간 동안 다른 스레드가 CPU를 사용할 수 있도록 하면 멀티스레드 사용이 보다 효과적이다. 그러나 이 보기의 경우처럼 홀수만 합하는 스레드와 짝수만 합하는 두 개의 스레드가 실행되는 시간의 합은 1에서 100000까지 더하는 시간뿐만 아니라 스레드의 문맥 교환에 따른 부가적인 시간이 소요되므로 하나의 스레드가 1에서 100000까지 더하는 경우보다 소요 시간이 커지게 되어 오히려 비효율적이다.

9. ② synchronized

10. ① 다른 스레드가 객체 a의 notify()를 호출할 때

13장 연습문제

이론 문제

1. ②
스트림은 다른 스트림과 연결될 수 있다.

2. (1) 동영상 파일(.avi) - 바이너리 파일이므로 바이트 스트림 클래스
(2) 메모장으로 작성한 파일(.txt) - 텍스트 파일이므로 문자 스트림 클래스
(3) 자바 클래스 파일(.class) - 바이너리 파일이므로 바이트 스트림 클래스
(4) HTML 파일(.html) - 텍스트 파일이므로 문자 스트림 클래스

3. ② FileReader
FileReader는 문자 스트림 클래스이다.

4. ② FileInputStream
동영상 파일을 읽으려면 파일 입출력 바이트 스트림 클래스를 사용해야 하므로 FileInputStream이 적당하다.

5. 총 5개의 문자를 읽어 출력하므로 다음과 같이 출력된다.

```
12345
```

6. "c:\\tmp\\sample.txt"의 경로명이 잘못되었거나 파일이 없는 경우에 FileNotFoundException 예외가 발생한다. try-catch 블록으로 묶으면 다음과 같다.

```
try {
   FileReader fin = new FileReader("c:\\tmp\\sample.txt");
}
catch (FileNotFoundException e) {
   System.out.println("파일을 찾을 수 없습니다.");
}
```

7.

```
FileOutputStream fout;
FileInputStream fin;
try {
   fout = new FileOutputStream("c:\\tmp\\test2.txt");
   fin = new FileInputStream("c:\\tmp\\test.txt");
   byte [] buf = new byte [128]; // 버퍼 할당
   while(true) {
      int n = fin.read(buf); // 버퍼 크기만큼 읽는다.
      fout.write(buf, 0, n); // 읽은 바이트만큼 쓴다.
      if(n < buf.length) // 버퍼 크기보다 적게 읽었다면
         break; // 파일 끝에 도달했으므로 복사 완료
   }
   fin.close();
   fout.close();
} catch (IOException e) { }
```

8. ① 파일 읽기

File 클래스는 파일을 읽거나 쓰는 기능은 제공하지 않는다.

9. (1) true

(2) "c:\Program Files\java\jre8" 문자열

(3) "c:\Program Files\java\jre8\Welcome.html" 문자열

(4) "Welcome.html" 문자열

(5)

```
File f = new File("c:\\Program Files\\java\\jre8", "Welcome.html");
```

10.

```
File a = new File("c:\\tmp\\a.txt");
if(f.exists()) System.out.println("yes");  // (1) c:\tmp에 a.txt 파일이 존재하면
                                                  yes 출력
System.out.println(f.length());            // (2) a.txt 파일 크기 출력
f.renameTo(new File("c:\\tmp\\b.txt"));    // (3) a.txt를 b.txt로 이름 변경
new File("c:\\tmp\\b.txt").delete();       // (4) b.txt 삭제
```