❑ 개념 확인

(1) 인터페이스 안에 정의할 수 선언을 모두 선택하세요.

1) public double PI = 3.14;
2) public double getValue();
3) public default double getValue() { return 0.0; };
4) static void hello() { System.out.println("Hello") }

(1,2,3,4)

(2) 인터페이스 안에 정의할 수 있는 메소드 선언을 모두 선택하세요.

1) private int getArea();
2) public float getValue(float x);
3) public void main(String [] args);
4) public static void main(String [] args);
5) boolean setValue(Boolean [] test);

(2,3,5)

(3) 다음 내용에 해당하는 Rectangle 클래스 정의의 첫 번째 문장만을 쓰세요.
"클래스 Rectangle은 IDrawable 인터페이스와 IMovable 인터페이스를 구현한다."

class Rectangle implements IDrawable, IMovable { }
class Rectangle implements IDrawable, IMovable { }
public class Rectangle implements IDrawable, IMovable { }
public class Rectangle implements IDrawable, IMovable { }

(4) 다음 내용에 해당하는 Audio 클래스 정의의 첫 번째 문장만을 쓰세요.
"Audio는 Sound를 확장하고(=상속받고) IMP3Play 인터페이스와 ITurnTablePaly 인터페이스를 구현한다."

class Audio extends Sound implements IMP3Play, ITurnTablePaly {
class Audio extends Sound implements IMP3Play, ITurnTablePaly
public class Audio extends Sound implements IMP3Play, ITurnTablePaly {
public class Audio extends Sound implements IMP3Play, ITurnTablePaly

(5) 다음 내용에 해당하는 IDrawable 인터페이스 정의의 첫 번째 문장만을 쓰세요.
"IDrawable 인터페이스는 IPaint 인터페이스를 상속 받는다."

interface IDrawable extends IPaint { }
interface IDrawable extends IPaint{ }
public interface IDrawable extends IPaint { }
public interface IDrawable extends IPaint{ }

(6) 다음의 인터페이스 정의에서 오류가 있으면 오류가 있는 문장을 올바르게 수정하세요.

```
public interface IMyInterface {
    void IMyMethod(int value) {
        System.out.println("인터페이스의 메소드 안 입니다.");
    }
}
```

default void IMyMethod(int value) { }
default void IMyMethod(int value){ }
public default void IMyMethod(int value) { }
public default void IMyMethod(int value) { }

(7) 다음과 같은 인터페이스와 클래스가 있다고 할 때, MyClass에서 반드시 구현해야 하는 메소드를 모두 고르세요.

```
 interface IA {
        public float mA(int a);
 }

 interface IB extends IA {
        public default int mB1(int a) { System.out.println("Here is mB1()"); };
        public Object mB2(int a);
 }

class C {
        public void mC(int a) { System.out.println("Here is mC()"); }
}

public class MyClass extends C implements IB {
}
```

1) mA()
2) mB1()
3) mB2()
4) mC()

1) mA(), 3) mB2()

(8) 다음의 인터페이스 선언과 사용에서 잘못된 점을 모두 지적하세요.

```
public interface IEdible {
        boolean amount;
        final int TYPE=10;
```

```
        public void eat() { };
};

public class Sandwitch extends IEdible {
        public void eat() { }
}

public interface Edible {
        //boolean amount; 필드는 정의할 수 없다.
        final int TYPE=10;
        public void eat(); // { };     추상 메소드만 정의할 수 있다.
};
public class Sandwitch implements Edible {
        public void eat() { }
}
```

(9) 클래스 Desk가 IMovable 인터페이스를 구현한다고 가정 했을 때, 다음의 문장들이 순차적으로 실행되면서 오류가 발생하는 문장을 찾고 이유를 설명하세요.

```
Desk desk = new Desk();
IMovable m = desk;
desk = m;
```

desk = m

(10) IControll 인터페이스를 아래와 같이 정의할 수 있습니다. IControll 인터페이스를 구현하는 IControllTest 클래스의 main() 메소드에서 익명 객체로 play()와 stop()을 작성하고 테스트 해 보세요.

```
public interface IControll {
      void play();
      void stop();
}
```

```
public interface IControll {
    void play();
    void stop();
}

public class IcontrollTest {
    public static void main(String arg[]) {

            IControll ic = new IControll() {
                    public void play() {
                            System.out.println("PLAY");
```

```
                }
                public void stop() {
                        System.out.println("STOP");
                }
        }; //
        ic.play();
        ic.stop();
    }
}
```

(11) 람다식에 대한 설명으로 틀린 것은 무엇인가?

① 람다식은 함수적 인터페이스의 익명 구현 객체를 생성한다
② 매개변수가 없을 경우 () -> {..} 형태로 작성한다
③ (a,b)->{return a+b; }는 (a,b)->a+b; 로 바꿀 수 있다
④ @FunctionalInterface가 기술된 인터페이스만 람다식으로 표현이 가능하다
⑤ 매개변수가 하나 일 경우 ()를 생략할 수 있다

(12) 중첩 멤버 클래스에 대한 설명으로 틀린 것은 무엇인가?

① 인스턴스 멤버 클래스는 바깥 클래스의 객체가 있어야 사용될 수 있다
② 정적 멤버 클래스는 바깥 클래스의 객체가 없어도 사용될 수 있다
③ 인스턴스 멤버 클래스 내부에는 바깥 클래스의 모든 필드와 메소드를 사용할 수 있다
④ 정적 멤버 클래스 내부에는 바깥 클래스의 인스턴스 필드를 사용할 수 있다

(13) 익명 객체에 대한 설명으로 틀린 것은 무엇인가?

① 익명 객체는 클래스를 상속하거나 인터페이스를 구현해야 생성될 수 있다
② 익명 객체는 필드, 매개변수, 로컬 변수의 초기값으로 사용할 수 있다
③ 익명 객체에는 생성자를 선언할 수 있다
④ 외부에서도 익명객체의 필드와 메소드에 접근할 수 있다
⑤ 익명객체에서는 부모클래스의 메소드를 재정의할 수 있다

(14) 다음과 같이 정의 된 Person을 상속받는 익명 객체를 생성하는 소스를 완성하고 테스트하세요.

```
class Person{
        void wake() {
                System.out.println("7시에 일어납니다");
        }
}
public class AnnoTest {
        public static void main(String[] args) {
                //익명 객체 생성
                Person person = [          ] {
                        //work() 메소드를 추가 합니다. 반환값은 없으며 "등교합니다"라는 문자열을 출력
                        [                                  ]

                        //부모클래스의 wake() 메소드 재정의하면서 work() 메소드 호출
```

```
                    }
                };

                //익명객체로 wake() 메소드 호출

                }
}
```