실력 상승! 연습 문제

Q1 다음은 추상 클래스 A를 상속해 클래스 B를 정의한 코드로, 오류를 포함하고 있다. 오류가 발생한 이유와 그 해결책을 쓰시오.

```
abstract class A {
    abstract void abc();
}
class B extends A {

}
```

오류가 발생한 이유	오류 해결책
클래스 B 내부에 추상메서드가 상속되어 포함되었지만 추상클래스로 정의하지 않음	해결책1. 클래스 B도 추상클래스로 정의함 (abstract class B {}) 해결책2. 추상메서드 구현 (오버라이딩)

Q2 다음과 같이 클래스 A와 클래스 B의 상속 관계가 있을 때 실행 코드(a.abc())의 결과가 "안녕하세요"가 나오도록 클래스 B 내부의 코드를 완성하시오.

```
abstract class A {
    abstract void abc();
}
class B extends A {
    @Override
    void abc() {
        System.out.println("안녕하세요");
    }
}
```

```
A a = new B();
a.abc();    // 안녕하세요
```

Q3 다음과 같은 추상 클래스 A가 정의돼 있다. 실행 코드가 "반갑습니다"를 출력하도록 익명 이너 클래스를 이용해 객체를 생성하는 코드를 완성하시오.

```
abstract class A {
    abstract void abc();
}

A a = new A() {
        @Override
        void abc() {
            System.out.println("반갑습니다");
        }
    };
a.abc();    // 반갑습니다.
```

Q4 추상 클래스의 객체를 생성하는 방법은 크게 다음과 같다. 각 방법의 장단점을 기술하시오.

① 추상 클래스를 일반 클래스로 상속해 객체 생성

② 익명 이너 클래스 사용

	장점	단점
①	여러 개의 객체 생성시 생성한 클래스의 생성자로 바로 생성 가능	추가 클래스를 정의하여 함
②	추가 클래스 정의가 불필요함	여러 개의 객체 생성시 동일한 오버로딩일 계속 작성해주어야 함

Q5 다음은 클래스와 인터페이스 간의 상속 문법이다. extends, implements, 불가능 중 적절한 상속 키워드를 넣으시오.

```
클래스 extends 클래스 {
    // ...
}
인터페이스 extends 인터페이스 {
    // ...
}

클래스 implements 인터페이스 {
    // ...
}

인터페이스 불가능 클래스 {
    // ...
}
```

Q6 다음과 같이 클래스 D가 인터페이스 A, B, 클래스 C를 상속하고자 할 때의 상속 문법을 완성하시오.

```
interface A {
}
interface B {
}
class C {
}
class D      extends C implements A, B      {
}
```

Q7 다음 코드는 인터페이스 A를 상속해 클래스 B를 정의한 코드로, 오류가 포함돼 있다. 오류가 발생한 이유와 해결책을 쓰시오.

```
interface A {
    void abc();
}
class B implements A {
    void abc() {
        // ...
    }
}
```

오류가 발생한 이유	오류 해결책

- 인터페이스 내부의 모든 메서드는 public abstract으로 정의됨 (미지정시 자동 추가).
- 상속하여 오버라이딩시 접근지정자는 좁아질 수 없음

B 클래스 내부의 abc() 메서드 앞에 public을 지정

Q8 다음 인터페이스 A는 디폴트 메서드를 포함하고 있다. 자식 클래스에서 부모 클래스의 abc() 메서드를 호출하는 코드를 추가해 다음과 같은 실행 결과가 나오도록 빈칸을 완성하시오.

```
interface A {
    default void abc() {
        System.out.println("A 인터페이스의 abc()");
    }
}
class B implements A {
    @Override
    public void abc() {

        A.super.abc();

        System.out.println("B 클래스의 abc()");
    }
}

B b = new B();
b.abc();
```

실행 결과

```
A 인터페이스의 abc()
B 클래스의 abc()
```