C\

시험범위

- 186p ~ 연습문제 위주로
- 연습문제 위치
  - 246~247 - Chapter 05 클래스의 기본
  - 312~314 - Chapter 06 메서드
  - 373~376 - Chapter 07 상속과 다형성
  - 402 - Chapter 08 클래스 심화
  - 444 - Chapter 09 인터페이스
  - 476 - Chapter 10 예외 처리
  - 506 - Chapter 11 델리게이트와 람다
  - 527~529 - Chapter 12 Linq
  - ~~프로젝트~~ - 안나온다

Chapter 05 - 클래스의 기본

연습문제 정답

문제 번호	정답	풀이
01	1: List - 클래스, list - 인스턴스	int도 클래스지만 자잘하게 보지 말자 **인스턴스는 new를 해야된다.**
02	3	메소드 안에서 클래스를 선언할 수 없다
03	`class Book{ string name; ...}`	클래스 선언 방식을 원하는거 같다.
04	`new Ramdom().Next()` 메소드	Next메소드는 3개가 있다. 194p 참고
05	직접 해보자	안해도 될 것 같다.

Chapter 06 - 메서드

문제 번호	정답	풀이
01	4	3. 인스턴스는 new를 할 때 생성된다. 4. 미리 정의되 클래스가 존재
02	4	반환형은 매개변수가 다를 때 달라도 된다. 즉 4번만 답.
03	같은 이름을 가진 메소드의 매개변수가 다른 것.	반환형이 다를 수 있으니
04	10, 40	만약 INT_MAX를 넘을 때는 long이 실행이 된다.
05	직접 해보자.	할 필요가 없어 보인다.
06	오버로딩 오류	오버로딩을 할 때는 매개변수가 달라야 하지만, 중복되는 매개변수가 있다.
07	A 생성자, B생성자, B 소멸자, A 소멸자	태어나는건 부모부터, 죽는건 자식부터

Chapter 07 - 상속과 다형성

문제번호	정답	풀이
01	부모 B, 자식 A	코드에서 A가 B를 상속받는다
02	오버로딩은 같은 메소드 이름에 다른 매개변수를 받는 것이고	오버라이딩은 자식이 부모의 메소드를 재정의 하는 것이다.
03	1	as키워드는 `exp is type ? (type)exp : (type)null` 와 같다.
04	2 -> 1 -> 3	`private`은 해당 클래스 내에서, `protected`는 상속받은 클래스 내, `public`은 모든 곳에서 된다.
05	20	하이딩 문제이다. 아래 참고
06	`((Parent)child).question`	하이딩 문제.
07	20	부모는 볼 필요가 없다. 캐스팅 하지않고 자식만을 사용한다.
08	20	위랑 같다. 그래도 명시적으로 부모를 하이딩 하였다.
09	20	위랑 같다. 부모는 virtual하였지만 자식이 하이딩 하였다.
10	20	위랑 같다. 오버로딩 하였지만 자식만을 사용한다.
11	O,O,O,O,X,X,X	6번은 ~~오버로딩~~, 오버라이딩이 안됨.
12	2	부모에 자식은 못담는다. 대신 자식에는 부모를 담을 수 있음.

섀도잉와 하이딩

> 개념이 매우 중요해서 따로 적는다.

- **섀도잉**

  C언어 기반의 프로그래밍 언어의 대부분은 이름이 겹칠 대 자신과 가장 가까운 변수를 사용한다.

```
class Program
{
    public static int number = 10;
    static void Main(string[] args)
    {
        int number = 20;
        Cnosole.WriteLine(number); // 20
    }
}
```

여기서 결과는 `20`이 나온다. 이유는 `int number = 20`이 가장 가까운 위치에 있기 때문이다.

이를 **섀도잉(shadowing)**이라고 부른다.

- **하이딩**

하이딩은 부모 클래스와 자식 클래스에서 일어나는 겹침이다.

```
class Parent
{
    public int variable = 273;
}
class Child : Parent
{
    public string variable = "shadowing";
}
main() { // 생략
    Child child = new Child();
    Console.WriteLine(child.variable); // shadowing
}
```

이 때 부모의 variable을 출력하고 싶으면 `((Parent)child).variable`로 출력하면 된다.

이를 조금 더 명시적으로 표시할 수 있는데, new 키워드와 override 키워드이다.

```
public new string variable = "shadowing"
```

이렇게 `new`를 넣으면 부모의 값은 무시(hide)를하고 자식에서 새로 만든다는 뜻이다. (`new`를 안쓴 것과 같다.)

```
Console.WriteLine((Parent)child.variable); // 273
```

다음은 오버라이드에 대해 알아보자. 먼저 부모에게 메소드가 오버라이드 될 거라는것을 명시 해줘야한다.

(**필드는 override 키워드와 virtual 키워드가 안된다.**)

```
public virtual int Do()
{
    return 1;
} // Parent class 내부
```

그리고 자식에서 `override` 키워드를 사용한다.

```
public override int Do()
{
    return 5;
} // Child class 내부
```

그리고 부모로 강제 형변환을 한 후 출력하면

```
Consle.WriteLine(((Parent)child).Do()); // 5
```

> virtual을 쓰지 않으면 overide에서 오류가 발생한다.

Chapter 08 - 클래스 심화

문제 번호	정답	풀이
01	2	반대이다.
02	`Question(out inputA, out inpubB);`	out키워드가 붙은 매개변수를 입력할 때 out키워드를 같이 사용해 줘야한다.

구조체 (struct)

구조체와 클래스의 차이

	클래스	구조체
정의	`class Pos {}`	`struct Pos {}`
인자값 없는 생성자	O	X
복사	참조 복사(포인터)	값 복사
생성자 규칙	어느 형태든 가능	모든 멤버 변수를 초기화 해야함.
맴버 변수의 초기화	선언과 동시에 초기화 O	선언과 동시에 초기화 X

Chapter 09 - 인터페이스

문제 번호	정답	풀이
01	3	인터페이스도 다형성을 사용 할 수 있다.
02	O,X,X,O,X,O	아래 참고

인터페이스 (참고용)

- 인터페이스에는 인스턴스 메서드와 속성만 넣을 수 있다.

  > 인스턴스 메서드 : `void Go(int a);`
  >
  > 속성 : `int data {get; set;}`

- 인터페이스에는 접근 제한자를 사용할 수 없다.

  > 접근 제한자: `public, private, protected` 등등..

- 인스턴스 메서드와 속성에 내부 구현은 할 수 없다.

```
int Cal(int a) // 에러
{
    return a + 1;
}
```

- 클래스는 **다중 상속**이 안되지만 인터페이스는 **다중 상속**이 가능하다.
- 인터페이스도 다형성이 가능하다.

```
IBasic basic = new TestClass();
```

- 인터페이스는 매너로 앞에 대분자 I를 항상 추가한다.

```
IDisposable, IComparable
```

Chapter 10 - 예외 처리

문제 번호	정답	풀이 (정답 2)
01	문법 오류는 프로그램이 컴파일이 안된다.	예외는 프로그램 실행 중 문제가 발생할 때 일어난다.
02	3	catch나 finally는 생략할 수 있다. 하지만 finally는 제일 마지막에 있어야 한다.
03	2	`throw Exception()` 으로 예외를 발생시킨다.
04	문법 오류 : 1, 4 / 예외 발생 2, 3	1: `new string[5];` 2: `FormatException` 발생 3: `IndexOutOfRangeException` 발생 4: `new Random();`
05	X, O, O, O, O, O	1번은 밑에서 다루겠다. 나머지는 책 참고

> try 구문에서 발생할 수 있는 예외의 종류가 다양하면, 이를 처리하기 위해 catch 구문을 여러번 사용해야한다?

책에는 이 문제에 대한 별다른 언급이 없다. 하지만 예외의 종류가 다양해도

```
catch (Exception e) {}
```

로 모든 예외를 받을 수 있다. `Exception` 클래스는 모든 예외의 부모이기 때문이다.

Chapter 11 - 델리게이터와 람다

문제 번호	정답	풀이
01	`delegate(매개변수, ..) {코드 ... return}`	`(매개변수, ..) => {코드 ... return}`
02	O, O, X, X	+: 델리게이터 추가, -: 델리게이터 삭제
03	3	클래스 밖에도 선언 가능. 메소드 안 불가능.

델리게이터와 람다식

람다식은 델리게이터 사용에 더 편리한 구문이다.

```
public delegate void OnClick(object sender); // 델리게이트 정의
// 사용 - delegate
OnClick = delegate(object sender) {
    Console.WriteLine("Clicked! Sender:" + sender);
}
// 사용 - 익명함수
OnClick = (sender) => {
    Console.WriteLine("Clicked! Sender:" + sender);
}
```

익명함수가 훨씬 간단해 보인다.

Chapter 12 - Linq

문제번호	정답	풀이
01	`output = from item in input where item < 4 select item`	예제에서 4 미만인 숫자를 리스트에 더한다.
02	`...input where item % 4 == 1 select item`	위 예제와 초반은 같지만 조건이 다르다
03	`...4 == 1 order by item select item`	문제 자체에서는 정렬이 된 상태라서 정렬이 필요 없지만 키워드가 있으니 적자.
04	`...order by item descending`	내림차순은 descending 키워드를 적는다.
05-1	`from item in products order by item.Name`	객체를 LinQ로 사용하려면 . 으로 접근한다.
05-2	`... where item.Price<2000 order by item.Price`	반대로 변환할 수 도 있어야 한다.
05-3	`...item.Price descending`	Reverse()를 사용했으므로 내림차순으로 한다.

상세 설명

> LInq란? `Language-Integrated Query`의 줄임말로 컬렉션 형태의 데이터를 쉽게 다루고자 SQL을 본따 만든 구문이다.

특징

- 모든 Linq질의는 **from, in, select**키워드를 포함해야 한다.

```
from <변수 이름(원하는 이름 지정)> in <컬렉션 이름>
select <결과에 넣을 요소>

// example
var result = from item in cars select item;
```

- 익명 객체

  C#에서는 클래스를 만들지 않아도 객체를 생성할 수 있다.

  ```
  new { id = 2, name = "홍길동" };
  ```

  이를 Linq랑 결합하면

  ```
  var output = from item in input where item < 50
               select new {
                   val = item,
                   powval = item * item
               };
  output // ex) [{val: 5, powval: 25}, {val: 4, powval: 16}]
  ```

  이런 문법이 가능하다.

- **Order By** 키워드

  Order By는 정렬을 하는 키워드인데 내림차순, 오름차순이 가능하다.

  - 오름차순 : `order by item [ascending] // 1 2 3 4 5` (ascending은 안적어도 자동으로 오름차순으로 된다.)
  - 내림차순 : `order by item [descending] // 5 4 3 2 1`

xml도 책에는 나오지만 문제에는 나오지 않아서 건너뛰겠다.

끗!