자바: 더 블랭크

2. 객체지향

객체지향으로의 이행

블랭크 1

- 점점 컴퓨터는 단순 계산이 아니라 실생활과 밀접하게 연관된 자동화 작업을 수행하게 되었다.

```java
package theblank.blank2;

import java.util.Scanner;

public class Blank21 {

	public static void main(String[] args) {
		String menus[] = {"빅맥", "상스치콤", "1955버거", "애플파이", "맥너겟"};
		String comments[] = {"우리가게 시그니쳐~~", "매콤한 치킨버거", "난빅맥보다이게더조아", "빼먹으면섭섭하자너", "가끔 쿠폰주면 사묵지"};
		int prices[] = {5000, 4000, 6000, 3000, 2500};

		System.out.println("지금은 한 1985년쯤... 이 프로그램은 미국맥도날드의 메뉴정보질의프로그램입니다..");
		System.out.print ("검색할 메뉴 이름을 입력해주세요. > ");

		Scanner sc = new Scanner(System.in);

		String query = null;
		while(!(query = sc.next()).equals("exit")) {
			for (int i=0; i<menus.length; i++) {
				String menu = menus[i];

				if (menu.equals(query)) {
					System.out.println(menus[i]);
					System.out.println(comments[i]);
					System.out.println(prices[i]);
					break;
				}
```

필 1

- 다룰 데이터가 많은 경우 이전과 같이 배열을 활용할 수 있다.
- 그러나 추가, 제거를 할 때 인덱스를 맞춰줘야 하는 등 데이터 관리에 있어 어려움이 많다.
- 항목이 수백여개라면... 어우 끔찍스

- 그런데, 실제 세계를 컴퓨터가 데이터로서 취급할 수 있다면? 이런 생각을 처음한 사람은 아마 무척 떨렸을 것이다. (사실 어떤 천재가 갑자기 이런 생각을 해낸 것은 아니다. 60년대 이후 점진적으로 OOP에 대한 연구가 발표된 것 같다.)
- 프로그램에서 다뤄야 할 실제 세계의 특정 부분을 객체로 나누어 파악하고 객체 단위로 설계한다. 작업 절차를 기준으로 프로그램을 설계했던 과거와는 다르다.

블랭크 2

메뉴를 객체 단위로 다룰 수 있도록 Menu 클래스를 만들고, 이전 프로그램을 수정해보세요.

```
package theblank.blank2;

class Menu {
        ----
}

class Blank22 {
        ----
}
```

필 2

- 클래스는 실제 세계의 객체를 메모리 상으로 옮기기 위한 설계도이다.
  - 클래스에는 객체와 관련된 속성을 한 곳에 모을 수 있다.
  - 뿐만 아니라 객체와 관련된 행위도 한 곳에 모을 수 있다.
- 클래스 자체는 설계도일 뿐이고, 우리는 클래스를 바탕으로 메모리 공간에 생성된 객체를 다루게 된다.
- 클래스로부터 객체를 생성하는 과정을 인스턴스화 과정이라고 하며, 이렇게 생성된 객체를 인스턴스라고 한다.
  - 객체라는 말에는 실제 세계의 객체를 컴퓨터 상에서 다룬다는 맥락이 내포되어 있다.
  - 인스턴스라는 말에는 클래스로부터 인스턴스화 과정을 통해 생성되었다는 맥락이 내포되어 있다.

블랭크 3

아래의 두 구문을 JVM이 실행하면 메모리 상에는 각각 어떤 일이 발생할지 그림으로 나타내보세요.

```
class BasicObject {
        //
        public BasicObject() {
                //
        }
        //
}

public class Blank23 {
        public static void main(String[] args) {
                int a = 123; // (a)
                BasicObject b = new BasicObject(); // (b)
        }
}
```

필 3

- (a) Primitive한 값은 해당 쓰레드의 스택 상에 올라간다. int 형은 4바이트(32비트)이므로 스택 상에 00000000000000000000000001111011이 할당된다.
- (b)
  - new BasicObject()
    - BasicObject의 인스턴스가 힙에 생성되고 BasicObject의 기본 생성자가 호출된다.
    - 완료 후 메모리 주소를 반환한다.
  - BasicObject b
    - 위의 연산에서 반환된 메모리 주소를 스택에 올린다. 앞으로 b에 접근하여 스택에 할당된 데이터로 접근할 수 있다.
  - 클래스가 하나의 사용자 정의 타입과 같이 기능하고 있음을 볼 수 있다.

객체지향 더 알아보기

블랭크 4

- 맥도날드는 세계에 여러 지점을 두고 있다.
- 블랭크 2에서 우리가 만든 메뉴 조회 프로그램에 세계 여러 지점의 메뉴을 통합하고자 하나, 이미 각 나라의 메뉴 데이터베이스의 컬럼명 구성이 상이하여 비용이 많이 발생한다.
- 이에 공통 컬럼을 menuName, menuComment, menuPrice로 강제하여 각 지부에 데이터베이스 수정을 요구하였다.
- 공통 컬럼을 속성으로 갖는 Menu 클래스와, 이를 상속하는 각 국가별 메뉴 클래스를 작성하여 프로그램을 수정해보세요.

필 4

- 상속: 조상 클래스를 포함하는 자식 클래스를 만든다.
  - 일반적으로 조상 클래스보다 자식 클래스의 속성이나 메서드가 더 많고, 의미적으로는 더 구체적으로 변한다.
  - IS-A: MenuKR과 MenuUS는 MenuCommon이다.
- 다형성: MenuCommon이라는 참조 변수로 MenuKR, MenuUS을 받았다.
- instanceof: 특정 객체의 클래스를 확인할 때 사용할 수 있는 연산자

블랭크 5

- MenuKR 'IS A' MenuCommon이라고 볼 수도 있지만, 메뉴 'HAS A' 현지화 정보라는 관계로 볼 수도 있다.
- 기본 메뉴 정보와 Localization이라는 현지화 정보를 필드로 갖는 Menu 클래스를 작성하여 앞선 프로그램을 수정해보세요.

필 5

- 자바에서는 단일 상속만 가능하기 때문에 무분별하게 상속 관계를 만드는 것은 주의해야 한다.
- 또한 현지화 정보를 필드로 갖도록 하는 방식이 상속하는 방식보다 확장성 측면에서 좋은 것을 볼 수 있다.
- static 메서드는 클래스가 클래스 로더에 의해 불러와지는 시점에서 메모리에 올라가며, 인스턴스가 아닌 클래스를 통해 바로 참조할 수 있다.

참고 자료 1
참고 자료 2
참고 자료 3

블랭크 6

- 각 메뉴에 대한 쿠폰 정보도 확인할 수 있게 만들고자 한다. 다만 쿠폰에 의한 가격 할인 방식은 각 메뉴마다 다르다.
- Menu의 필드에 추상 클래스 Coupon을 주입받아 이 기능을 구현해보세요.
  - 25% 할인 쿠폰을 빅맥에 적용
  - 한국에서만 사용할 수 있는 1000원 할인 쿠폰을 불고기버거에 적용

필 6

- 추상 클래스는 추상 메서드를 가지고 있는 클래스이다. 추상 메서드는 메서드의 선언부만 있고, 실제 바디는 기술되지 않은 메서드를 말한다.
- 이러한 추상 클래스는 그 자체로는 사용할 수 없고, 상속을 통해 추상 메서드를 오버라이딩 하는 방식으로 사용할 수 있다.
- 즉, 클래스가 설계도라면, 추상 클래스는 미완성 설계도이다.

블랭크 7

- 블랭크 6의 Coupon 클래스를 인터페이스로 변경해보세요.

필 7

- 인터페이스는 특수한 추상 클래스로 이해할 수 있다. 다만 abstract하지 않은 일반적인 속성과 메서드도 담을 수 있는 추상 클래스보다 제약 조건이 많아 더 높은 추상화 정도를 가진다고 할 수 있다.
  - 인터페이스의 모든 속성은 public static final
  - 인터페이스의 모든 메서드는 public abstract
  - 제어자는 생략할 수 있다.