به نام خدا

تمرین چهارم درس برنامه نویسی پیشرفته

نام و نام خانوادگی: امیرحسام ادیبی نیا

شماره دانشجویی: ۹۹۳۱۰۸۷

ترم زمستان ۰۰ - ۹۹

سوال اول

- **الف)**
  - Polymorphism
    - چندریختی یا Polymorphism در شئ‌گرایی، به اشیا این امکان را می‌دهد که چند نوع مختلف را اختیار کنند. به طور دقیق‌تر، هر شئ از یک کلاس امکان اختیار کردن نوع همان کلاس و یا تمام زیر کلاس‌ها را دارد.
  - Substitution
    - در اصل Substitution، بیان شده است که اگر در هر جایی از برنامه که شئ‌ای از کلاس پدر مورد استفاده قرار گرفته است، کلاس فرزند را قرار دهیم، در اجرای برنامه نباید مشکلی پیش بیاید.
  - Abstract class
    - کلاس Abstract، کلاسی است که دارای یک یا چند متد پیاده‌سازی نشده است و تمام کلاس‌های فرزند آن باید آن متدها را پیاده‌سازی کنند تا برنامه امکان اجرا شدن داشته باشد.
  - Interface
    - Interface را می‌توان کلاسی کاملاً Abstract دانست (البته Interface را نمی‌توان یک کلاس دانست). در واقع Interface مجموعه‌ای از متدهای پیاده‌سازی نشده است که کلاس‌هایی که از آن Interface استفاده می‌کنند، باید حتماً آن‌ها را پیاده‌سازی کنند.
- **ب)**

۱. غلط است؛ زیرا علاوه بر این که از لحاظ منطقی درست نیست، در کد هم به مشکل می‌خوریم. برای مثال امکان Override کردن متد را نخواهیم داشت و کامپایلر توانایی تشخیص دو متد را از هم نخواهد داشت.

۲. غلط است. کامپایلر تنها هنگامی خودش تابع کانستراکتور کلاس پدر را صدا می‌زند که در کلاس پدر، کانستراکتوری بدون هیچ آرگومانی وجود داشته باشد.

۳. صحیح است. هنگام ارث‌بری، کانستراکتورهای کلاس پدر، در کلاس فرزند به ارث نمی‌رسند.

۴. صحیح است. متدهای Override شده نمی‌توانند دسترسی کمتری داشته باشند. اما برا متدهای Overload شده، محدودیتی وجود ندارد.

۵. غلط است. در جاوا متدهای private و static را نمی‌توان Override کرد.

۶. صحیح است. یک کلاس می‌تواند یک و یا چند Interface را پیاده‌سازی کند.

- **ج)**
  - Overloading کردن متد یک کلاس به این معناست که متد دیگری با همان نام، اما با ورودی و یا خروجی مختلف تعریف کنیم. اما Override کردن، متدی از یک کلاس فرزند را بر روی متد پدر آن تعریف می‌کنیم. به عبارتی دیگر، Overloading در یک کلاس اتفاق می‌افتد، اما Overriding در دو کلاس فرزند و پدری اتفاق می‌افتد.

سوال دوم

- **الف)**
  - از آنجا که می‌توان یک شئ از جنس فرزندان را به شئ‌ای از پدر آن کلاس نسبت داد، در نتیجه تنها خط اول و سوم درست هستند. زیرا برای مثال هر Husky ای یک Animal هست اما هر Mammal ای یک Cow نیست.
- **ب)**
  - می‌دانیم عملیات a = b را در صورتی می‌توانیم انجام دهیم که Dynamic Type این دو شئ یا با هم برابر باشند و یا Dynamic Type شئ a جزو اجداد شئ b باشند. با این توضیح تنها عملیات به صورت زیر خواهد بود:

```
m2 = m1; // Mammal = Mammal : Correct
m1 = d2; // Mammal = Dog : Correct
d2 = d1; // Dog = Dog : Correct
a3 = a2; // Animal = Animal : Correct
a1 = m1; // Animal = Mammal : Correct
a2 = m2; // Animal = Mammal : Correct
a3 = h1; // Animal = Husky : Correct
m2 = h1; // Mammal = Husky : Correct
a1 = c1; // Animal = Cow : Correct
```