گزارش‌کار پروژهٔ عید - برنامه‌سازی پیشرفته (جاوا)

پاسخ سوالات قدم دوم

پیشوا آزیز - 40313003

پاسخ سوال ۱

copy Shallow و copy Deep به مفاهیم مربوط به کپی کردن آبجکت‌ها برمی‌گردد. وقتی سعی می‌کنیم آبجکتی را کپی کنیم و آن‌را در جایی دیگر ذخیره کنیم باید در مورد این موضوع یعنی نوع کپی کردن آگاهی داشته باشیم. copy Shallow را با یک مثال بهتر متوجه می‌شویم:

فرض کنید کلاسی با نام `Person` داریم که در آن یک فیلد از نوع `Address` که آن هم خود یک کلاس است، وجود دارد.

```
class Address {
    public String city;
    public String country;

    public Address(String city, String country) {
        this.city = city;
        this.country = country;
    }
}

class Person {
    public String name;
    public Address address;

    public Person(String name, Address address) {
        this.name = name;
        this.address = address;
    }

    public Person copy() {
        return new Person(this.name, this.address);
    }
}
```

شاید این‌گونه به نظر برسد که متد copy می‌تواند آبجکتی جدید از نوع Person بازگرداند و مشکلی در پیاده‌سازی آن وجود ندارد. در حالی که با تست کردن عملکرد آن می‌توان متوجه شد که مشکل آن چیست. حال با ساختن یک آبجکت از نوع Person و کپی کردن آن در آبجکتی دیگر با استفاده از متد copy کدی مانند زیر خواهیم داشت:

```
public class Main {
    Address location = new Address("Hewler", "Kurdistan");
    Person person1 = new Person("Hozan", location);
    Person person2 = person1.copy();

    person2.name = "Dilan";
    person2.address.city = "Amed";

    System.out.println(person1.name + " is from " + person1.address.city);
    System.out.println(person2.name + " is from " + person2.address.city);
}
```

با مشاهدهٔ خروجی این کد، می‌توان فهمید تغییری که در یکی از فیلدهای person2 انجام شد، باعث همین تغییر در person1 نیز شده است. البته این فیلد از نوع Address بوده و به دلیل این که رفرنس آن از person1 به person2 کپی شده است، تغییرات یکسانی داشته‌اند؛ وگرنه تغییری که در person2.name اعمال شده است، موجب تغییر در person1.name نشده است، چون فیلد name از نوع primitive است. لذا آبجکت‌ها و رفرنس‌های داخلی هر آبجکت همان‌گونه باقی مانده‌اند و نسخه‌های جدیدی از آن‌ها به‌وجود نیامده است. این نوع کپی کردن که در آن آبجکت کپی‌شده، با همان رفرنس‌های داخلی قبلی کپی می‌شود، Shallow copy می‌باشد.

این موضوع در این‌جا، با توجه به مشکلی که در پایان قدم اول پروژه مشاهده شد، در اصل نوعی عملکرد نامطلوب دیتابیس است، لازم است تا متد مربوط به کپی کردن آبجکت‌ها به گونه‌ای صحیح پیاده‌سازی شود تا رفرنس‌هایی که در آبجکت‌ها مورد استفاده قرار گرفته است، متمایز از هم باشند و به یک فضای واحد در حافظه اشاره نکنند.

با این توصیف، می‌توان پیش‌بینی کرد که Deep copy چه تمایزاتی دارد. Deep copy به نوعی از کپی کردن آبجکت‌ها اشاره می‌کند که در آن، از رفرنس‌ها و آبجکت‌های داخلی هم کپی گرفته می‌شود و به‌جای هر کدام از آن‌ها، نسخه‌های جدیدی در آبجکت کپی‌شده قرار می‌گیرند. برای این منظور می‌توان پیاده‌سازی متد copy را به شیوهٔ زیر تغییر داد:

```
public Person copy() {
    Person copied = (Person) super.clone();
    copied.address = new Address(this.address.city, this.address.country);
    return copied;
}
```

همان‌گونه که مشاهده می‌شود، نسخهٔ جدیدی از فیلد address که از نوع reference می‌باشد، ایجاد شده است. با این نوع پیاده‌سازی، دیگر هرگونه تغییر در هر یک از فیلدهای آبجکت‌های person1 یا person2، تاثیری بر دیگری نمی‌گذارد.

پاسخ سوال ۲

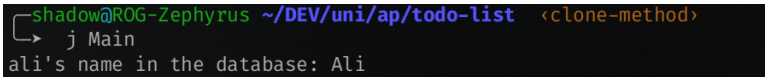
در اینجا به‌خاطر یکی از قابلیت‌های جاوا به نام Covariant return types، می‌توان نوع خروجی از متدها را تغییر داد، به شرطی که موارد زیر برقرار باشند:

۱. نوع خروجی از متد overrideکننده، باید یک زیرکلاس از نوع خروجی متد overrideشده باشد.

۲. این تغییر سیگنچر فقط برای reference typeها پاسخ‌گو است و برای primitive typeها امکان‌پذیر نمی‌باشد.

با توجه به موارد بالا، چون کلاس Human زیرکلاسی از کلاس Entity است، می‌توان نوع خروجی متد copy را از Entity به Human تغییر داد.

پاسخ سوال ۳



اسکرین‌شات از خروجی کد حین اجرای بخش تست کد

پاسخ سوال ۴

متد `clone` در جاوا برای ایجاد کپی از یک آبجکت استفاده می‌شود. این متد در کلاس `Object` تعریف شده است. برای استفاده از آن شرایطی مانند implement کردن اینترفیس `Cloneable` توسط کلاس مورد نظر (برای جلوگیری از اکسپشن مربوطه) و override کردن متد `clone` و فراخوانی `super.clone()` وجود دارد. استفاده از این متد به تنهایی، آبجکت مورد نظر را صرفا Shallow copy می‌کند و برای پیاده‌سازی Deep copy باید به صورت دستی رفرنس‌های موجود در داخل آبجکت‌ها را کپی کرد.

پاسخ سوال ۵

مطابق سورس‌کد خود جاوا، اینترفیس `Cloneable` هیچ متدی در تعریف خود ندارد و استفاده از آن برای کلاسی که متد `clone` را پیاده‌سازی می‌کند صرفاً جنبهٔ مارکر دارد و نشان می‌دهد که کلاس ذکرشده از cloning پشتیبانی می‌کند.
از طرفی ضرورت استفاده از این اینترفیس می‌تواند به مفاهیم clean code به‌خصوص واضح‌تر شدن و جلوگیری از عملکرد غیرمنتظره و نامطلوب مربوط شود.