**تکلیف شماره 2**

**تمرین شماره 1**

**الف)** در هر شئ State به معنای ویژگی و مشخصاتی که دارد اشاره میکند، برای مثلا برای یک شئ از یخچال، اندازه، مصرف انرژی، نوع کابری و ... دارد. Behavior در هر شئ به عملگر ها و امکانات هر شئ اشاره میکند برای مثال یک یخچال میتواند دمای محتویات درون خود را پایین بیارد یا مثلا دمای داخلش را روی مانیتورش نشان بدهد و در آخر Identity در هر شئ به هویت آن شئ اشاره میکند، برای مثال یک یخچال نمی تواند یک تلویزیون یا یک اتومبیل باشد.

**ب)**

**تمرین شماره 2**

1. غلط. کامپایلر در صورتی که کانستراکتوری به صورت دستی نوشته شده باشد، کانستراکتور پیش فرضی ارائه نمیدهد.
2. صحیح. ArrayList به دلیل اینکه یک سری فعل و انفعالات برای چک کردن مرز های آرایه و تداخل های تصادفی را انجام میدهد کمی کند تر از array معمولی است.
3. غلط. همه ی کلاس های کالکشن در جاوا مقادیری از آبجکت ها نگهداری میکنند که در آن primitive type ها جای نمی گیرند، برای استفاده مشابه از آنها میتوان از کلاس های Integer, Boolean, Double, Byte و ... استفاده کرد.
4. صحیح. برای هر کلاس میتوان کانستراکتور های متفاوت و متعددی با پارامتر های ورودی متفاوتی ایجاد کرد (به اصطلاح Constructor Overloading).
5. صحیح. در متد و کانستراکتور های هر کلاس استفاده از کیوورد this به مقادیر متغیر های همان کلاس اشاره دارد.
6. صحیح. اگر به یک متغیر لوکال مقداردهی اولیه داده نشود برنامه دچار ارور کامپایل خواهد شد.
7. غلط. اگر در یک متد یک متغیر خاصی تعریف شود به آن local variable میگوییم و فقط در همان متد قابل دسترسی هستند.
8. صحیح. اگر در کانستراکتور مقداردهی ای انجام شود، مقدار های اولیه کلاس تغییر پیدا می کنند.

**تمرین شماره 3**

**الف)**

1. وقتی تابعی را اورلود میکنیم پارامتر های ورودی آن باید با تابع parent متفاوت باشد، که در این صورت داشتن تابع اورلود شده (که پارامتر های آن متفاوت است) با return type متفاوت امکان پذیر است. ولی اگر پارامتر ها یکسان باشند همچین چیزی امکان ندارد و خطای کامپایل ایجاد میکند.
2. از کانستراکتور برای مقدار دهی اولیه یک شئ استفاده می شود در حالی که از متد برای نمایش توابع و functionality یک شئ استفاده می شود. کانستراکتور ها نمی توانند چیزی را برگردانند (return type ندارند) ولی متد ها باید ریترن تایپ مشخصی داشته باشند. در صورت عدم وجود کانستراکتور، کامپایلر به صورت پیشفرض یک کانستراکتور استفاده میکند که همچین چیزی برای متد ها صادق نیست. کانستراکتور حتما باید همنام با کلاس باشد در صورتی که متد میتواند هر نام دلخواهی باشد. کانستراکتور به صورت ضمنی و خودکار توسط سیستم صدا زده می شود در صورتی که متد توسط برنامه نویس و کاربر صدا زده میشوند.
3. بله، یک کلاس میتواند چند کانستراکتور با پارامتر های ورودی متفاوت باشند. برای اورلود کردن کانستراکتور باید یک کانستراکتور مانند کانستراکتور قبلی ساخت اما با پارامتر های ورودی متفاوت. در اورلود کردن کانستراکتور ها میتوان از کیوورد this() استفاده کرد، که با قرار دادن پارامتر ورودی در پرانتز، به کانستراکتوری که چنین ورودی هایی میگیرد برنامه را ارجاع میدهیم.

**ب)** اولین ایراد در خط اول تابع main هستش که به کانستراکتور کلاس person فرمت متناقض پاس داده شده، برای رفع این مشکل باید در کد کانستراکتور کلاس person در گرفتن id به جای int از String استفاده کنیم.

ایراد دیگر در همان بخش این است که برای کانستراکتور کلاس person ، return type مشخص شده که باید برداشته شود. (به طور کلی کانستراکتور return type ندارد)

در خط اول کلاس person نام Firstname از قاعده camelCase پیروی نمی کند و در نتیجه باید به firstName تغییر پیدا کند.

در خط دوم کلاس person نیز نام last\_name از قاعده camelCase پیروی نمی کند و باید به lastName تغییر پیدا کند.

به دلیل تغییری که در ورودی کانستراکتور ایجاد کردیم در خط سوم کلاس person نیز باید تایپ id را String کنیم.

سپس به دلیل درست کردن نام گذاری متغیر ها برای اینکه در کانستراکتور اختلالی ایجاد نکند باید قبل از هر 3 متفیر ( نام، نام خانوادگی، و آیدی) کیوورد this گذاشته شود مثل this.firstName

در تابع express که یک ورودی String میگیرد، return type تابع از نوع اینتجر تعریف شده که با توجه به نوع استفاده از آن، وجود آن ضروری نبوده و میتوان آن را به void تغییر داد.

در تابع print باید تغییراتی که در نامگذاری ایجاد کرده ایم رو دوباره انجام دهیم.

\*\* به دلیل تغییر نکردن فیلد های نام، نام خانوادگی و id و immutable بودن آنها میتوان آنها را از نوع final تعریف کرد.

public class Main **{** public static void main**(**String**[]** *args***) {** person p1 = new person**(**"Ted", "Mosby", "123456"**)**;  
 p1.express**()**;  
 p1.express**(**"Happy"**)**;  
 p1.print**()**;  
 **}** static class person **{** private final String firstName;  
 private final String lastName;  
 private final String id;  
  
 public person**(**String *firstName*, String *lastName*, String *id***) {** this.firstName = *firstName*;  
 this.lastName = *lastName*;  
 this.id = *id*;  
 **}** public void express**() {** System.out.println**(**"I feel neutral"**)**;  
 **}** public void express**(**String *state***) {** System.out.println**(**"I feel " + *state* + " today"**)**;  
 **}** public void print**() {** System.out.println**(**"person{" +  
 "name='" + firstName + '\'' +  
 ", lname='" + lastName + '\'' +  
 ", id=" + id +  
 '}'**)**;  
 **}  
 }  
}**