سوال سوم - الف

ج۱ : method overloading به معنای کنترل کردن قطعه کد اجرایی توسط compiler توسط شرایط فراخوانی تابع است . java یکی از اولین زبان هایی است که method overloading را معرفی کرد و jvm این کار را به این صورت انجام می دهد که وقتی یک تابع صدا میشود ورودی های آن را از نظر تعداد و نوع آنها آنالیز می کند ؛ و بعد در لیست method های تعریف شده تابعی را که با ان هماهنگی کامل دارد را فراخوانی می کند و اگر به هر دلیلی تابعی که دقیقا با آن همخوانی ندارد با پیدا نکند Exception را throw می کند. حال برای اینکه خود java عملیات overloading را بتواند انجام دهد (به این منظور که بداند که کدام تابع ها overload شده ی یک تابع دیگر هستند تا بتواند بعدا در هنگام صدا سازی تابع عملیات مقایسه را انجام دهد) نیاز به method signature دارد که در زبان java مفهوم return type برخلاف C جزی از ان نیست و پس نتیجه میگیریم که وقتی return type را عوض می کنیم در واقع jvm هیچ فرقی برایش نمی کند و overloading خود را به درستی انجام میدهد.

ج۲ : در برنامه نویسی oop چندین نوع تابع داریم که یکی از ان دسته ها magic function ها هستند. این تابع ها معمولا توسط برنامه نویس صدا زده نمی شوند(هر چند در بعضی از زبان ها این امکان وجود دارد ولی باز هم در اکثر آنها bad programming practice تلقی می شود.) یکی از این magic method ها همان constructor ها هستند. constructor ها در واقع توابعی هستند که در هنگام ساختن instance variable از روی یک class ؛ حتی اگر در خود class یک constructor یی به صورت implicit تعریف نشده باشد ؛ توسط jvm صدا زده می شوند و دیگر برنامه نویس هیچ کنترلی درباره ی زمان اجرا شده آنها ندارد. این ویژگی به برنامه نویس این اجازه را می دهد که دقیقا و بدون هیچ نگرانی مطمئن باشد که دقیقا ان کدی که در تعریف کرده است هنگام ساختن instance variable از class ش انجام داده می شود و اصطلاحا برنامه نویس می تواند یک default behaviour برای شیا خود در شرایط درست کردن نمونه ی جدید تعریف کند. از طرفی دیگر method ها صرفا تابع هایی هستند که هیچ معنایی ندارند تا اینکه از آنها استفاده شود و استفاده کردن از آنها به عهده ی برنامه نویس است و در واقع به برنامه نویس این توانایی را می دهد که behaviour های اضافه تری به object ش بدهد .

ج۳: همانطور که در بالا هم اشاره شد در واقع constructor خودش یک نوع method به اسم magic method است و در واقع نوع خاصی از method ها یا همان توابع است در زبان java است . حال اگر ما چند تابع با اسم خود class داشته باشیم که نوع ورودی های انها با هم متفاوت است (چون constructor اصلا return type معناداری ندارد دیگر نیاز نیست نگران ان باشیم) jvm به این صورت عمل می کند که با توجه به نوع ورودی ها و تعداد آنها مقایسه انجام می دهد و ان constructor یی را صدا می زند که همخوانی کاملی با ورودی ها دارد و به این صورت عملیات

overloading را handle می کند و همچنین اگر در طی فرایند مقایسه اگر هیچ method یی را به نیابد که به صورت کامل با آن همخوانی دارد یک exception می دهد.

سوال سوم - ب

نکته : این نکته حائز اهمیت است که اشکالاتی که در نام گذاری گرفته می شود مقداری به نظر خود شخص خواننده ی کد هم برسی دارد و من در این code review نظر خودم را کاملا اعمال می کنم و چیز هایی که لزوما اشتباه نیستند را اشتباه در نظر میگیرم.

در این code review سعی میکنم مشکلات را به صورت بالا به پایین لیست کنم که برای شخص برسی کننده هم مشکلی ایجاد نکند.

- در خط اول می بینیم که که برای نام class از person استفاده شده است که با p نوشته شده است اما نکته ی مهم اینجاست که بهتر است اسم class ها را به صورت pascal case نوشت و آن را باید به صورت Person تغییر نام داد.
- باز هم در خط اول می بینیم که برای اسم متغیر از p1 استفاده شده است که ترجیح من به این صورت است که از person1 یا person استفاده شود و از اسم های تک حرفی که اصلا مشخص نمی کنند که هدف از ساخت آنها چیست دوری شود.
- بازهم در خط اول میبینیم که ورودی هایی که به constructor method داده شده است سه تا string است اما هنگامی که به تعریف آن در خود class مراجعه می کنیم متوجه میشویم که constructor برای id انتظار یک int را دارد
- در خط دو به نظر من عملیات overloading که در اینجا برای احساسات شخص انجام شده است بی معنی است اگر کسی کد را بخواند نمی تواند هیچ جوره تشخیص بدهد که اگر به express هیچ ورودی ندهد class از خودش چه behaviour یی نشان می دهد.
- در نکته ی بعدی متوجه میشویم که class به صورت static درون همان فایل تعریف شده است و به نظر من این کار کاملا اشتباه است و من خودم شخصا تقریبا هیچ class را به صورت static در درون یک class دیگه تعریف نمی کنم مگر اینکه واقعا اینکار معنی دار باشد اما در اینجا صرفا کد را شلوغ و بی معنی می کند.
- در خط اول تعریف class به اسم person داریم که FirstName با F نوشته شده است و در صورتی که good practice به این صورت که با camelcase نوشته شود و ان را به صورت firstName بنویسند.
- در خط دوم تعریف می بینیم که last_name به صورت snake case نوشته شده است اما همانطور که در مورد قبلی اشاره شد بهتر است که به صورت lastName نوشته شود.
- در مورد بعدی به constructor و این موضوع که return type دارد میرسیم و این کاملا اشتباه است و jvm در هنگام اجرای این قطعه کد هیچگاه آن را به عنوان constructor برای خودش تفسیر نمی کند و به عنوان یک simple method آن را تعریف می کند.
- در مورد بعدی به اسم تابع express میرسیم که برای من زیاد مشخص کننده ی هدف ان نبود و اگر اسم آن را به expressFeeling تغییر می دادیم بهتر بود.
- در مورد بعدی به return type اضافه ای که برای express می رسیم و می بینیم که در یک مورد void است و در یک مورد دیگر نیز int است و این کار باعث می شود که فرایند overloading انجام نشود و حالا در صورتی که کد اصلا درست هم اجرا شود (بستگی به jdk که از ان استفاده می کنیم جواب های نتیجه های مختلفی شاید بگیریم) تابع ها ان گونه که ما میخواهیم رفتار نمی کنند.

- در مورد بعدی به اسم ورودی برای تابع express میرسیم که به نظر من state نام درستی نبود و اگر feeling بود خیلی بهتر بود هر چند این ایراد خیلی سخت گیرانه است و بیشتر بر اساس ایرادی است که از اسم تابع در بالاتر گرفتم و پیشنهاد دادم که آن را به expreesFeeling تغییر دهیم.
- در مورد بعدی به return 0 می رسیم که در بالا هم اشاره کردم باید از پاک شود و return type به void تغییر پیدا کند.
- نکته ی بعدی در تابع print است که ترجیح خود من این است که برای دسترسی به propertyهای class م اگر در static method نیستم از this استفاده کنم.
- نکته ی بعدی هم که می توان اشاره کرد این است که private field ها هیچ کدام نه getter و setter یی ندارد. هر چند که گذاشتن getter یا setter به هیچ وجه ضروری نیست اما به نظر می رسد که به احتمال زیاد در آینده به id یا firstName یک تابع نیاز پیدا خواهیم کرد پس بهتر بود که حداقل getter برای انها می گذاشتیم و از طرفی اگر id برای یک person قرار است هیچ گاه تغییر داده نشود آن را final می کردیم.
-