سوال اول:

قسمت اول: توضیح دهید چرا نمی توان عضوهای یک collection را با استفاده از foreach حذف کرد؟

می دانیم در foreach خودش از عنصر 0ام collection شروع به iterate کردن میکند و تا سایز آن کالکشن منهای یک پیش میرود و مدیریت میکند که به طور پیوسته از ایندکس 0 محتوای آن داخل متغیر element ساخته شده ،ریخته شود.

```
for (Element Type element : collection){
        Loop body
}
```

پس foreach بیشتر زمانی مفید است که بخواهیم تمام آیتم های اون collection را بررسی کنیم

وقتی از متد remove استفاده میکنیم و یک عضو از collection را حذف میکنیم index مربوط به اعضا آپدیت میشود. وقتی از while برای پیمایش استفاده میکنیم خودمان به ایندکس ها دسترسی داریم و باید با دقت آن ها را کنترل کنیم.که برای این کار جاوا Iterator را پیاده سازی کرده و همه ی collection ها یک آبجکت از این جنس دارند که مدیریت اینکس ها را بر عهده میگیرد.

با foreach نمیتوان از متد remove استفاده کرد چرا که کد مربوط به کنترل و آپدیت ایندکس ها و عوض شدنشان برای آن در جاوا نوشته نشده ودر صورت انجام این کار با خطای concurrent modification exception روبرو میشویم و نمیتوان همزمان با این که روی عناصرکاکشن حرکت میکنیم، عناصر آن را حذف کنیم.

قسمت دوم: متد ()main از کدام access modifier باید استفاده کند؟ چرا؟

می دانیم تابع main در جاوا یک تابع استاندارد است که به وسیله JVM برای شروع اجرای هر برنامه ی جاوایی استفاده میشود.

access modifier برای تابع main باید به صورت public باشد. زیرا در این صورت برای سایر کلاس ها حتی آن هایی که با این کلاس در یک package نیستند ؛ در دسترس و قابل رویت میباشد. اگر سطح دسترسی به صورت public نباشد ،کلاس های JVM نمی توانند به آن دسترسی داشته باشند و در اجرای برنامه مشکل پیش می آید.

قسمت سوم: .قاعده ی information hiding را توضیح دهید.

در علوم کامپیوتر information hiding یک قاعده برای تفکیک قالب و تصمیمات طراحی یک برنامه کامپیوتری که در معرض تغییر هستند؛ میباشد تا اگر یک تصمیم طراحی تغییر کند سایر قسمت های برنامه دچار تغییرات وسیع نشود. به عبارت دیگر، information hiding جلوگیری از دسترسی client ها به بخش‌هایی از کلاس یا کامپوننت نرم‌افزاری است و این کار یا از طریق امکاناتی که زبان برنامه‌نویسی در اختیار برنامه‌نویس قرار می‌دهد مانند (private variable)یا ... انجام می‌گیرد. در خیلی از موارد Encapsulation به جای information hiding به کار میرود.

Encapsulation قاعده ی خصوصی ساختن فیلدها در یک گروه ایجاد دسترسی به فیلدها از طریق متودهای عمومی می باشد. اگر یک فیلد خصوصی اعلام شود، در دسترس هیچ کس دیگر در خارج گروه قرار نمی گیرد. به این دلیل encapsulation با عنوان مخفی کردن داده (information hiding) نیز مطرح می شود.
Encapsulation به عنوان یک مرز محافظتی توصیف می شود که مانع دسترسی به کد و داده توسط دیگر کدهای تعریف شده در خارج گروه می شود. دسترسی به داده و کد به شدت توسط یک اینترفیس کنترل می شود.
مزیت مهم encapsulation توانایی اصلاح کد اجرا شده بدون شکستن کد دیگری که از کد ما استفاده می کنند، می باشد.

سوال دوم:

الف) درستی یا نادرستی عبارات زیر را همراه با توضیح مختصری تعیین کنید

۱. اگر constructor کلاس A از نوع private تعر یف شده باشد، فقط داخل خود کلاس A می توان از آن instance ساخت .

درست- همان طور که میدانیم constructor نیز نوعی تابع است که اگر به صورت private تعریف شود فقط در آن کلاس میتوان به آن دسترسی داشت و از این رو فقط در خود کلاس میتوان instance ساخت.

۲. فیلد هایی که private تعریف شده اند ، داخل package خود قابل دسترسی اند .

نادرست – فیلد هایی که private تعریف شده اند ، داخل کلاس خود قابل دسترسی اند. اگر قبل از فیلد سطح دسترسی را تعریف نکنیم آنگاه آن فیلد داخل packageخود قابل دسترس است و در واقع سطح دسترسی protected دارد.

۳. می توان تعداد محدودی آرگمان ورود ی به ()main داد .

نادرست- بی شمار آرگمان میتوان به عنوان ورودی main داد و به کمک []args از جنس String میتوان n تا آرگمان را به برنامه پاس داد. سپس میتوان در جاوا آنها را به هر primitive type نیز تبدیل کرد.

ب) جاهای خالی را با عبارت مناسب پر کنید.

۱. با استفاده از **تعریف فیلدها به صورت static**، میتوان به فیلدهای یک کلاس بدون ساخته شدن یک شی از آن کلاس دسترسی داشت.

۲. تابع length() **اندازه ی یک آرایه(تعداد المنت های آرایه)** و size() **اندازه ی** یک ArrayList را خروجی می دهند.