به نام خدا

تمرین سوم درس برنامه نویسی پیشرفته

نام و نام خانوادگی: امیرحسام ادیبی نیا

شماره دانشجویی: ۹۹۳۱۰۸۷

ترم زمستان ۰۰ - ۹۹

سوال اول

- **الف)**
  - Wrapper class
    - کلاس‌های Wrapper، کلاس‌هایی است که به منظور استفاده‌ی نوع‌های ابتدایی[1] به عنوان شئ پیاده‌سازی شده‌اند. برای مثال، برای داشتن شئ‌ای از نوع int می‌توان از کلاس Integer استفاده کرد.
  - Autoboxing & Unboxing
    - تبدیل و یا کست کردن یک نوع ابتدایی به کلاس Wrapper آن نوع را Autoboxing و عکس این عملیات را Unboxing می‌گویند.
  - Garbage Collection
    - جاوا برای مدیریت حافظه‌ی خود، ابزاری با نام Garbage Collector دارد که وظیفه‌ی آن در اجرای برنامه، بررسی حافظه‌ها و آزاد کردن حافظه‌هایی است که برنامه دیگر با آنها نیاز ندارد. به این فرآیند، Garbage Collection می‌گویند.
- **ب)**

۱. خیر، Garbage Collection تنها حافظه‌هایی که مورد استفاده قرار نمی‌گیرند را آزاد می‌کند. ولی برای مثال ممکن است در برنامه‌ای تعداد زیادی شئ ساخته شود و رفرنس‌های آنها مورد استفاده قرار گیرند و آنها پاک نشوند. در نتیجه‌ی این کار ممکن است برای اجرای برنامه حافظه کم بیاید.

۲. هنگامی که در جاوا، متغیری از نوع یک کلاس تعریف می‌کنیم، تنها یک رفرنس ساخته می‌شود. به عبارتی دیگر، در حافظه پشته[2]، اشاره‌گری از جنس آن کلاس تعریف می‌شود. اما به هیچ

[1] Primitive Types
[2] Stack Memory

شئ‌ای اشاره نمی‌کند. پس از آنکه یک شئ از جنس آن کلاس ساخته شد، حافظه‌ای در هیپ[3] برای آن شئ تخصیص پیدا می‌کند و پوینتر، به آن اشاره می‌کند.

۳. روش‌ها و کد مربوطه:

روش اول: می‌توان با حلقه‌ی for این کار را انجام داد.

روش دوم: می‌توان با حلقه‌ی while این کار را انجام داد.

روش سوم: می‌توان با استفاده از iterator این کار را انجام داد.

روش چهارم: می‌توان با استفاده از حلقه‌ی for-each این کار را انجام داد.

```
ArrayList<Integer> a = new ArrayList<Integer>();
a.add(1);
a.add(2);
a.add(3);

for (int i = 0; i < a.size(); i++)
   System.out.println(a.get(i));

int index = 0;
while (index < a.size()) {
   System.out.println(a.get(index));
   index++;
}

for (int i : a)
   System.out.println(i);

Iterator<Integer> it = a.iterator();
while (it.hasNext())
   System.out.println(it.next());
```

۴. ارور Stack Overflow هنگامی رخ می‌دهد که حافظه‌ی پشته به هر دلیلی محدودیت خود را رد کند. برای مثال زیاد صدا زدن توابع بازگشتی و یا تعریف کردن تعداد زیادی شئ می‌تواند باعث بروز این خطا شود. برای جلوگیری از بروز این خطا می‌توان علاوه بر اصلاح منطق کد،

[3] Heap Memory

محدودیت حداکثری سایز حافظه پشته را زیاد کرد. برای این کار، لازم است هنگام کامپایل کردن برنامه، با استفاده از آرگومان Xss- محدودیت حافظه پشته را افزایش داد.

۵. کلاس HashMap همانطور که از اسمش معلوم است، مانند یک تابع عمل می‌کند و مجموعه‌ای از کلیدها[4] را با مجموعه‌ای از مقدارها[5] متناظر می‌کند. کلاس HashSet اما، خود یک مجموعه را ذخیره می‌کند. به این معنا که عضو تکراری ندارد و ترتیب را مانند HashMap حفظ نمی‌کند.

[4] Keys
[5] Values

سوال دوم

- ۱)
  - صحیح است. کپسوله‌سازی[6]، میزان وابستگی[7] در برنامه را کاهش می‌دهد.
- ۲)
  - غلط است. Package Access تنها برای کلاس‌های داخل یک پکیج قابل دسترسی است.
- ۳)
  - غلط است. کلاس‌ها، تنها می‌توانند public و یا default باشند.
- ۴)
  - غلط است. Map ها، مانند تابع عمل می‌کنند و هر کلید را با یک مقدار، متناظر می‌کنند.
- ۵)
  - غلط است. در جاوا، متغیرها همیشه Pass by value می‌شوند. به طور دقیق‌تر، در جاوا رفرنس‌هایی که به مقدار اشاره می‌کنند، ساخته می‌شوند. در نتیجه اگر مقدار رفرنس‌ها را تغییر دهیم، مقدار اشاره شده تغییر نمی‌کند.
- ۶)
  - صحیح است. در جاوا می‌توان Constructor های private تعریف کرد.

[6] Encapsulation
[7] Dependency

سوال چهارم

- **الف)**
  - جدول حافظه در انتهای برنامه به شکل زیر خواهد بود:

Memory Address	Heap Memory	Field Name
1000	c1	Coordinate
1001	0	x
1002	0	y
1003	c2	Coordinate
1004	3	x
1005	0	y
1006	c3	Coordinate
1007	0	x
1008	4	y
1009	c4	Coordinate
1010	1	x
1011	1	y
1012	c5	Coordinate
1013	3	x
1014	0	y
1015	t1	Triangle
1016	1000	c1
1017	1003	c2
1018	1006	c3

Memory Address	Stack Memory	Field Name
100	1000	c1
101	1003	c2
102	1006	c3
103	1009	c4
104	1012	c5
105	1015	t1
106	6	area
107	12	perimeter

- **ب)**
  - می‌دانیم که عملگر == بررسی می‌کند که دو رفرنس داده شده، به یک حافظه‌ی یکسانی اشاره می‌کنند یا نه. اما متد equals، بررسی می‌کند که آیا دو حافظه‌ای که رفرنس‌ها به آنها اشاره می‌کنند مقدار یکسانی دارند یا خیر. حال جاوا برای رشته‌هایی که هنگام تعریف مقداردهی می‌شوند (مقداردهی از نوع رشته‌ی ثابت و نه با استفاده از متد) و مقدار یکسانی دارند، یک حافظه‌ی مشترک در هیپ تخصیص می‌دهد و در صورتی که هر دو رفرنسی را با هم مقایسه کنیم، جواب مثبت خواهد بود. در نتیجه، دو خط اول خروجی مقدار true چاپ می‌شود. اما از آنجا که رشته‌ی str4 به شکل گفته شده ساخته نشده است، هنگامی که با رشته‌ی str3 با عملگر == مقایسه می‌شود، پاسخ منفی خواهد بود. ولی از آنجا که مقدار یکسانی دارند، پاسخ مقایسه‌ی دوم مثبت خواهد بود. در نتیجه خروجی به صورت زیر خواهد بود:

```
true
true
false
true
```