به نام خدا

محمد رضا حسن زاده **۹۷۳۱۷۰۳** **تمارین برنامه نویسی پیشرفته**

سوال اول:

۱- کلاسها کتگوری یا به عبارتی نوع یک شی را مشخص می‌کنند. کلاس (Class) در برنامه نویسی شئ گرا، نوعی قالب برنامه نویسی قابل گسترش است که برای ایجاد اشیاء، تعیین مقادیر اولیه‌ی وضعیت و مشخص نمودن رفتار آنها مورد استفاده قرار می‌گیرد. در واقع، کلاس نقشه‌ی نوعی و مشترک برای گروهی از اشیاء است که ویژگیهای مشترکی داشته، و رفتارهای مشترکی از خود نشان می‌دهند. یعنی کلاسها انواعی هستند که شخص برنامه نویس، خود می‌تواند، آنها را برای حل مسئله‌های دنیای واقعی طراحی کند. یک کلاس، یک مفهوم بسط یافته از ساختمان (structure) است که به جای این که، فقط داده ها را نگهداری کند، می‌تواند هم داده ها و هم توابع را با هم نگهداری کند.

تفائت آن با شی این است که شی به عبارتی می‌توان گفت که زیر مجموعه‌ی کلاس است زیرا شی به عنوان یک نمونه از کلاس است و از رفتار هایی که داخل آن کلاس برایش تعریف شده است تبعیت می‌کند. شی نمونه‌ای از کلاس است.

۲- سازنده های یک کلاس یا به عبارتی constructor های یک کلاس تایپ خروجی ندارند و این وجه تمایز بین تابع و سازنده می‌باشد. به هنگام ساختن یک شی از آن کلاس باید داخل پرانتز های نام کلاس فیلد هایی که در کلاس و سازنده تعریف شده است را مقدار دهی اولیه کرد. اما برای صدا زدن توابع باید با استفاده از . (dot) به توابع دسترسی پیدا کرد و متد ها را روی این شی صدا زد و این هم یک تفاوت متد ها با سازنده است.

۳- کدی که برای مقدار دهی آن استفاده شده است ایراد دارد و باید به صورت زیر اصلاح شود.

```
public class Book{
   private String title;
   public Book(String title){
      this.title = title;
   }
}
```

همانطور که در کد بالا مشاهده می‌کنید ابتدا باید اسم خود سازنده با خود کلاس یکی باشد و همچنین برای مقدار دهی برای فیلد title چون هم نام هستند هم فیلد و هم ورودی سازنده باید برای مقدار دهی فیلد از کلمه کلیدی this به همراه یک dot استفاده کنیم.

سوال دوم:

الف) غلط است. زیرا سازنده وقتی تعریف می‌کنیم و می‌خواهیم که شی‌ای از این کلاس بسازیم به هنگام درست کردن این نمونه حتما باید آرگومان های ورودی سازنده مقدار دهی شود وگرنه با خطا مواجه می‌شویم.

ب) غلط است. سطح دسترسی کلاس می‌تواند اگر تعریف نشود به صورت default-package باشد البته در صورت نیاز اینگونه تعریف می‌شود.

پ) غلط است. زیرا وقتی protected تعریف شوند علاوه بر خود class نیز subclass ها هم می‌توانند از این متغیر ها استفاده بکنند.

ت) غلط است. متغیر های داخل توابع مربوط به اسکوپ خود توابع هستند. به عبارتی یک متغیر local هستند و تنها در داخل اسکوپ خود می‌توان به آن ها دسترسی پیدا کرد.

سوال سوم:

۱- Primitive type ها طبق جدول زیر می‌باشند

Data types	Size	Range and Description
byte	1 bytes	-128 to 127
short	2 bytes	-32768 to 32767
int	4 bytes	-2^{31} to $2^{31}-1$
long	8 bytes	-2^{63} to $2^{63}-1$
float	4 bytes	1.40129846432481707e-45 to 3.40282346638528860e+38 مثبت و منفی
double	8 bytes	4.94065645841246544e-324d to 1.79769313486231570e+308d مثبت یا منفی
boolean	1 bytes	true or false
char	2 bytes	یک کاراکتر یا حرف از مقدار های کد اسکی

short , byte , int , long : یک عدد صحیح در بازه‌ی بالا که ذکر شده است در خود ذخیره می‌کنند

float: یک عدد اعشاری تا ۶ یا ۷ رقم اعشار طبق بازه‌ی بالا در خود ذخیره می‌کند

double: یک عدد اعشاری تا ۱۵ رقم اعشار طبق بازه‌ی بالا در خود ذخیره می‌کند

boolean , char : در جدول بالا ذکر شده است

۲- مفهوم overloading به این معناست که مثلا می‌توان دو سازنده ساخت با آرگومان های ورودی متفاوت یعنی مثلا دو سازنده ساخت که تمامی موارد آن ها یکسان باشد ولی در آرگومان های ورودی خود متفاوت باشند. مانند قطعه کد زیر که در آن مفهوم overloading انفاق افتاده است.

```
public Test(){
   this.t = 0;
}
public Test(int t){
   this.t = t;
}
```

در مثال بالا مشاهده می‌شود که هر دو سازنده نام و visibility modifier یکسان دارند اما در ورودی ها متفاوت اند.

البته این موضوع تنها در سازنده ها نیست. به طور مثال با متد ها هم می‌توان این موضوع را پیاده سازی کرد.

۳- با توجه به اصل encapsulation باید توجه داشته باشیم که فیلد های مربوط به کلاس باید به صورت private تعریف شوند تا از دسترس دیگران و همچنین تغییر دادن این فیلد ها در امان باشند. در نتیجه برای دسترسی به مقدار داخل یک فیلد یا تغییر مقدار یا ... از یک سری متد هایی استفاده می‌کنیم به نام ها getter و setter که این متد ها به ترتیب مقدار فیلد را به ما خروجی می‌دهند و مقدار فیلدی را می‌توان با آن ها تغییر داد البته این موضوع کاملا به وسیله‌ی شخص برنامه نویس کنترل می‌شود تا اصل encapsulation نیز رعایت شود. به طور مثال به قطعه کد زیر توجه کنید

```
public class Test {
   private int test;


   public void setTest(int test) {
      this.test = test;
   }

   public int getTest() {
      return test;
   }
}
```

همانطور که دیده می‌شود توابع setTest و getTest برای دسترسی و تغییر مقدار داخل فیلد استفاده می‌شوند