سوال اول

آ)

کلاس یک طرح برای شی است. می توانیم کلاس را به عنوان یک طرح (نمونه اولیه) یک خانه تصور کنیم. این شامل تمام جزئیات در مورد طبقات ، درب ، پنجره ها و غیره است. بر اساس این توصیفات خانه را می سازیم. خانه شیء است.

ب) این تابع مقداری را بر نمی‌گرداند.

پ)

- نام کلاس باید به Book تغییر کند.
- نام constructor باید با حرف بزرگ شروع شود.
- متغیر title در سمت چپ تساوی باید به صورت this.titleباشد.
- در سمت چپ تسوای ; کم است.

ت)

تابع static مستقل از شی است و بدون ساختن شی از کلاس میتوان از آن استفاده کرد و در رم یک فضای ثابت برای کد اشغال می‌کند ولی تابع غیر ایستا نیاز به شی برای صدا زده شدن دارد و به ازای هر شی ایجاد شده کد آن در حافظه کپی می‌شود.

متغیر کلاسی استاتیک نیز مستقل از شی است و بدون ساخت شی نیز در حافظه قرار می‌گیرد و همانند تابع استاتیک یکتا است و مقدار آن در تمامی اشیاء از یک جنس یکسان خواهد بود و همچنین GB هیچ گاه آن را پاک نمی‌کند. ولی متغیر کلاسی غیر استاتیک در هر شیء، مجزا ساخته می‌شود و تا زمانی که آن شیء مورد استفاده باشد پاک نمی‌شود.

سوال دوم

الف)

1. public
2. New
3. Inside class, Inside package, outside package subclass, outside package

ب)

1. غلط، اولین کلمه با حروف کوچک شروع شود
2. غلط، این متغیر ها local variable هستند و فقط در متدی که در آن تعریف شده اند قابل استفاده هستند.
3. غلط، متغیرهای instance primitive به طور پیش فرض مقداردهی اولیه می شوند در صورتی که به هر متغیر محلی باید صریحاً یک مقدار نسبت داده شود.
4. صحیح
5. غلط، در صورت تعریف شدن constructor کامپایلر دیگر constructor دیفالتی نمیسازد.
6. صحیح
7. غلط، یک آرایه نمی تواند انواع مختلفی از مقادیر را ذخیره کند.

ج)

1. برای آرایه‌ای که اعضای آن از نوع primitive باشند:

متد در واقع کپی از آن عضو را گرفته و تغییرات را بر روی نسخه کپی شده ایجاد میکند، در نتیجه عضو در آرایه اصلی تغییری نمیکند. البته اگر رفرنسی به کل آرایه (و نه فقط یک عضو خاص) به متد داده می‌شد تغییرات ایجاد شده روی اعضا در داخل متد با آرایه خارج از متد هم منعکس می‌شد.

برای آرایه‌ای که اعضای آن از نوع primitive نباشند (اعضا از نوع reference type باشند):

با توجه به اینکه متد کپی از رفرنسی آن عضو را دریافت می کند تغییراتی که بر روی آن ایجاد می کند بر روی عضو در آرایه اصلی نیز منعکس می‌شود.

For individual primitive-type elements of an array: A called method receives and manipulates a copy of the value of such an element, so modifications do not affect the original value. If the reference of an array is passed to a method, however, modifications to the array elements made in the called method are indeed reflected in the original. For individual elements of a reference type:A called method receives a copy of the reference of such an element, and changes to the referenced object will be reflected in the original array element.

2. بله ولی در صورتی که تایپ پارامترها حتما متفاوت باشند. زیرا اگر تنها return type تغییر کند ولی تایپ پارامترها و اسم متدها یکسان باشند، overloading محسوب نمیشود و با compile error مواجه میشویم.
باید توجه شود که در overloading ، متدها باید نام یکسان ولی پارامترهای ورودی متفاوتی داشته باشند .

3. هنگامی که یک رشته با کمک یک عملگر انتساب(=) و به کمک literal شکل می‌گیرد، وارد String constant pool می‌شود، و به همین علت اگر مجددا رشته ای با محتوای یکسان تولید کنیم رشته جدید نیز به همان محتوای قبلی در String constant pool اشاره میکند.

4. برای بررسی این موضوع می توانیم از قطعه کد زیر استفاده کنیم با اجرای این قطعه کد مشاهده میکنیم که خروجی true خواهد بود:

```
public bool checking() {
        String first = "InterviewBit";
        String second = "InterviewBit";
        if (first == second)
                return true;
        else
                return false;
}
```

در صورتی که هنگام ساخت رشته به کمک new() به هر رشته یک حافظه جداگانه تخصیص داده می‌شود و حتی اگر دو رشته با محتوای یکسان داشته باشیم دو شی جدا برای آنها در نظر گرفته می‌شود. در واقع در این حالت تابع checking، false برمیگرداند.

```
public bool checking() {
        String first = new String("InterviewBit");
        String second = new String("InterviewBit");
        if (first == second)
                return true;
        else
                 return false;
}
```

When a String is formed as a literal with the assistance of an assignment operator, it makes its way into the String constant pool so that String Interning can take place. This same object in the heap will be referenced by a different String if the content is the same for both of them.

```
public bool checking() {
        String first = "InterviewBit";
        String second = "InterviewBit";
        if (first == second)
                return true;
        else
                return false;
```

```
}
```

The checking() function will return true as the same content is referenced by both the variables.

Conversely, when a String formation takes place with the help of a new() operator, interning does not take place. The object gets created in the heap memory even if the same content object is present.

```
public bool checking() {
        String first = new String("InterviewBit");
        String second = new String("InterviewBit");
        if (first == second)
                return true;
        else
                return false;
}
```

The checking() function will return false as the same content is not referenced by both the variables.