سوال اول

الف)

1. درست
2. نادرست

ب)

1. Primitive
2. Set
3. Anonymous
4. ArrayList
5. Static

ج)

1. تفاوت ArrayList و LinkedList:

LinkedList	ArrayList
از doubly linked list برای ذخیره داده‌ها استفاده می‌کند	از dynamic array برای ذخیره داده‌ها استفاده می‌کند
هم List و هم Queue را implement می‌کند	فقط List را implement می‌کند
برای دسترسی به داده مورد نظر، باید لیست را پیمایش کرده تا آن را پیدا کنیم؛ در نتیجه کندتر خواهد بود	دسترسی به داده مورد نظر به کمک اندیس بوده و سریع‌تر است
حذف کردن یک المان سریع‌تر است؛ کافیست تا next عنصر قبلی را تغییر دهیم	حذف کردن یک المان کندتر است زیرا نیاز به شیفت دادن بقیه اعضا خواهیم بود
مناسب برای کنترل داده (حذف، اضافه و ...)	مناسب برای مرتب‌سازی و دسترسی به داده

2. در جاوا می‌توانیم متدهای همنام داشته باشیم؛ با این شرط که حتما در نوع، تعداد و یا ترتیب پارامترهای ورودی متفاوت باشند (توجه داشته باشید که تفاوت در return type اجباری نیست). به این کار method overloading می‌گویند.
حالت‌های مختلف آن عبارت اند از:

الف) تغییر دادن **تعداد** پارامترها

```
private static void sayHi(){
    System.out.println("Hello!");
}

private static void sayHi(String name){
    System.out.println("Hello " + name + "!");
}
```

ب) تغییر دادن **نوع** پارامترها

```
private static void log(int x){
    System.out.println("I got an int!");
}

private static void log(String x){
    System.out.println("I got a String!");
}
```

ج) تغییر دادن **ترتیب** پارامترها

```
private static void display(String name, String id){
    System.out.println("name= " + name + " id= " + id);
}

private static void log(String id, String name){
    System.out.println("id= " + id + " name= " + name);
}
```

3. در جاوا، حذف کردن یک عضو از لیست با استفاده از foreach ممنوع است و java.util.ConcurrentModificationException رخ خواهد داد. دلیل آن این است که پیمایش لیست، همزمان با حذف کردن یک آیتم از آن (تغییر دادن لیست به طور کلی‌تر)، در حال انجام است.

راه‌حل این مشکل، استفاده از Iterator است:

```
List<String> names = ....
Iterator<String> it = names.iterator();
while (it.hasNext()) {
    String name = it.next();
    // Do something
    it.remove();
}
```

4. کالکشن هش‌مپ متشکل از مجموعه‌ای از keyها و valueهاست. در جاوا به کمک ۳ متدی که در کد زیر آمده‌اند، می‌توانیم به

- مجموعه keyها
- مجموعه valueها
- مجموعه جفت‌های (key,value)

دسترسی داشته باشیم و روی آن‌ها پیمایش کنیم.

```
// iterating through (key,value) mappings
System.out.println("Entries: ");
for(Entry<String, String> entry: myHashMap.entrySet()) {
   System.out.print(entry);

// iterating through keys
System.out.println("Keys: ");
for(String key: myHashMap.keySet())
   System.out.println(key);

// iterating through values
System.out.println("Values: ");
for(String value: myHashMap.values())
   System.out.println(value);
```

سوال دوم

الف)

از آن جایی که این لیست دائم در حال تغییر است و نیاز به جستجو میان داده‌ای نداریم (و تنها نیاز به پیمایش آنها به ترتیب اضافه شدنشان به لیست داریم) استفاده از LinkedList برای این سناریو مناسب است.

ب)

از آن جایی که می‌خواهیم یک لیست از اسامی دانشجویان ذخیره کنیم و می‌خواهیم نام هر نفر حداکثر یکبار در لیست ظاهر شود استفاده از HashSet برای این سناریو مناسب است.

ج)

از آن جایی که لیست دانشجویان دانشگاه تغییر زیادی ندارد اما نیاز داریم هر روزه بارها در این لیست جستجو کنیم استفاده از ArrayList برای این سناریو مناسب است.

د)

از آن جایی که حداکثر ظرفیت هر کلاس مشخص است و میدانیم استفاده از Collection های جاوا نسبت به استفاده از آرایه‌ها overhead زیادی دارد، در چنین سیستمی با حافظهٔ محدود استفاده از آرایه بهتر است.

ه)

در این سناریو میتوان از یک HashMap استفاده کرد. جستجو در HashMap ها به دلیل استفاده از تکنیک hashing بسیار سریع انجام میشود، بنابراین می‌توانیم به عنوان key ها شمارههای دانشجویی و به عنوان value ها شیء دانشجوی مربوطه را ذخیره کنیم و از این طریق با جستجوی شماره‌های دانشجویی به سرعت به دانشجوی مورد نظر دسترسی داشته باشیم.