آزمون پایانی برنامه نویسی پیشرفته

زمان امتحان: ۱۲۰ دقیقه

مدرس: صادق علی اکبری

تاریخ: شنبه ۲۲ خرداد ۱۳۹۵

جمع نمرات بیشتر از ۱۰۰ است. پس روی سؤالاتی که مسلط هستید تمرکز کنید. به وقت امتحان دقت کنید.

۱. (۱۶ نمره) هر یک از سؤالات زیر را در یک خط (حداکثر دو خط) پاسخ دهید:

الف) دو تفاوت Set و List چیست؟

ب) دو تفاوت interface و abstract class را ذکر کنید.

ج) متد hashCode چه کاربردی دارد؟ یک مثال بزنید.

د) تفاوت کلاس‌های ServerSocket و Socket چیست؟

۲. (۶ نمره) قطعه کد زیر نحوه استفاده از کلاس MyGeneric را نشان می‌دهد. کلاس MyGeneric را تعریف کنید. راهنمایی: این کلاس فقط شامل یک فیلد (value) و یک متد (invoke) است. برای سادگی، بدنه متد invoke را هم خالی بگذارید. پارامتر دوم متد invoke همیشه یک رشته (String) است و پارامتر اولش هم‌نوع متغیر value است.

```
MyGeneric<Integer> g1 = new MyGeneric<Integer>();
g1.value = 12;
g1.invoke(11, "Ali");
MyGeneric<String> g2 = new MyGeneric<String>();
g2.value="A";
    g2.invoke("B", "Ali");
```

۳. (۱۵ نمره) یک متد با نام serialize بنویسید که آدرس یک فایل و همچنین یک شیء را به عنوان پارامتر بگیرد و شیء موردنظر (پارامتر دوم) را در فایلی با آدرس موردنظر (پارامتر اول) ذخیره (serialize) کند و در نهایت حجم (طول) فایل ساخته شده را برگرداند.

مثال: قطعه‌برنامه زیر شیئی از نوع Person را در فایلی به نام d:\serial1.bin ذخیره می‌کند و مقدار ۹۳ را چاپ می‌کند (زیرا طول این فایل ۹۳ بایت است). سپس در فایلی با نام c:\serial2.bin شیئی از نوع String ذخیره می‌کند و مقدار ۱۶ را چاپ می‌کند (طول فایل c:\serial2.bin برابر با ۱۶ بایت خواهد بود)

```
 Person p = new Person("Ali", "Alavi");
long fileLength = serialize("d:\\serial1.bin", p);
System.out.println(fileLength);

System.out.println(serialize("c:\\serial2.bin", "Ali Alavi"));
```

۴. (۱۰ نمره) خروجی اجرای برنامه زیر چیست؟ این برنامه خطای کامپایل ندارد. فقط خروجی را ذکر کنید، توضیح لازم نیست.

```
class MyException extends Exception { }
class YourException extends RuntimeException { }
public class Exceptions {
  public static void main(String[] args) {
    try {
      f();
      System.out.println("After f method");
      System.out.println(1 / 0);
    } catch (MyException e) {
      System.out.println("catch MyException");
    } catch (YourException e) {
      System.out.println("catch YourException");
    } catch (Exception e) {
      System.out.println("catch Exception");
    }finally{
      System.out.println("finally");
    }
    System.out.println("after");
  }

  private static void f() throws Exception {
    System.out.println("f method");
    throw new MyException();
  }
}
```

۵. (۱۵ نمره) واسط MapUtils را پیاده‌سازی کنید (کلاسی بنویسید که این واسط را implement می‌کند)

```
interface MapUtils{
     Map<String, Integer> createMap(List<String> list, Set<String> blackSet);
}
```

نحوه پیاده‌سازی متد createMap : یک جدول (map) برمی‌گرداند که هر سطر از این جدول شامل یک رشته و یک عدد است. هر رشته‌ای که در list باشد ولی در blackSet نباشد، یکی از کلیدهای این جدول خواهد بود (دقت کنید که list و blackSet پارامترهای متد createMap هستند). مقدار متناظر با هر کلید هم تعداد تکرار آن رشته در list است.

مثلاً اگر list شامل رشته‌های "Naghi", "Naghi", "Naghi", "Taghi", "Ali", "Ali" و blackSet شامل رشته‌های "Vali" , "Naghi" باشد، جدول حاصل که متد createMap برمی‌گرداند این‌گونه خواهد بود:

Ali	2
Taghi	1

٦. (٢٠ نمره) برنامه زیر را در نظر بگیرید:

```
Singer shajarian = new Singer("Mohammadreza Shajarian");
shajarian.play(2);
Musical musical = shajarian;
musical.play(2);
Person person = shajarian;
System.out.println(person.getName());
Nameable nameable = shajarian;
System.out.println(nameable.getName());
musical = new Piano();
musical.play(2);
```

خروجی این برنامه عبارت است از:

```
Haaay: 2
Haaay: 2
Mohammadreza Shajarian
Mohammadreza Shajarian
Ding: 2
```

می‌دانیم دستور زیر یک خطای کامپایل (syntax error) دارد:

```
Nameable n = new Piano();
```

همچنین می‌دانیم کلاس Person به این صورت پیاده شده است:

```
class Person implements Nameable{
	private String name;
	@Override
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
}
```

الف) شما Singer ، Musical ، Nameable و Piano را به صورت کامل پیاده‌سازی کنید (فقط هر چه لازم است پیاده کنید. دقت کنید که از این موارد، برخی interface و برخی class هستند)

ب) نمودار UML Class Diagram را برای این موارد ترسیم کنید.

٧. (٥ نمره) در یکی از مشاغل، بسیاری از افراد دچار معضلاتی مانند زانودرد، کمردرد، درد گردن، افتادگی شانه‌ها و ضعف در بینایی می‌شوند.

الف) به نظر شما این افراد چه شغلی دارند؟ ١. کارگر معدن ٢. بوکسور ٣. زورگیر ٤. مهندس کامپیوتر

ب) به نظر می‌رسد این مشکل در الگوی غلط کار در این افراد است. کدام گزینه زیر برای بهبود الگوی کار در این افراد مفید است؟

١- الگوی صحیح نشستن هنگام کار ٢- قطع کار حداقل پنج دقیقه در هر ساعت و سپس راه رفتن، نگاه به افق و نرمش صحیح ٣- هر دو مورد

ج) آیا ورزش مستمر و منظم هم تأثیری در این مشکلات دارد؟

۸. (۱۶ نمره) متد f را در نظر بگیرید. این متد مشخصات یک فرد را بررسی می‌کند (اصطلاحاً validate می‌کند).

```
boolean f(String firstName, String lastName, int age, String email) throws Exception{
  if(firstName==null || firstName.length()==0)
    throw new Exception("Bad Format");
  if(lastName==null || lastName.length()==0)
    throw new Exception("Bad Format");
  if(email==null || email.length()==0)
    throw new Exception("Bad Format");

  if(!email.contains("@"))
    return false;
  if(firstName.length()>20)
    return false;

  return true;
}
```

الف) این کد شامل حداقل سه نوع Bad Smell است. کد فوق را Refactor کنید (کد بازنویسی‌شده را بنویسید).

ب) برای تست کد بازنویسی‌شده، unit test بنویسید (با کمک JUnit). ضمناً این تست باید موارد زیر را بیازماید:

– برای فردی با مشخصات «علی، علوی، ۲۵ و ali@alavi.ir » این متد مقدار true برگرداند.

– برای فردی با ایمیل نامعتبر (مثلاً AAA) ، مقدار false برگرداند.

– برای فردی با نام‌خانوادگی null این متد یک خطا پرتاب کند.

نکته: برای صرفه‌جویی در وقت، در این سؤال ذکر import ها لازم نیست. همچنین اگر در کد بازنویسی شده کلاس جدیدی ایجاد می‌کنید، در تعریف این کلاس ذکر getter ها و setter ها و constructor ها لازم نیست.

۹. (۱۵ نمره) کلاس Processing را در نظر بگیرید. متد process ، متد اصلی این کلاس است که قرار است روی کامپیوتری با حداقل چهار هسته پردازشی اجرا شود. این متد، متد Util.*timeConsuming* را فراخوانی می‌کند که دارای حجم زیاد محاسبات است (کند است). با کمک برنامه‌نویسی چندنخی، کلاس Processing و متد process را طوری تغییر دهید که خروجی متد process تغییر نکند (البته ترتیب چاپ خروجی‌ها در متد process مهم نیست) ولی برای لیست‌های بزرگ، سریعتر اجرا شود. برای سادگی فرض کنید تعداد اعضای list مضربی از چهار است. در صورت نیاز می‌توانید متدها و کلاسهای دیگری نیز تعریف کنید.

```
class Processing{
  public static void process(List<String> list) {
    for (String element : list)
      if(Util.timeConsuming(element))
        System.out.println(element);
  }
}
```

رمضانتان مبارک، نماز و روزه‌تان قبول. دلتان شاد.

علی‌اکبری