آزمون پایانی درس برنامه‌نویسی پیشرفته

سه‌شنبه ۲۳ خرداد ۱۳۹۱

مدت امتحان: ۱۲۰ دقیقه

مدرس: صادق علی‌اکبری

---

۱. درباره Unicode و UTF8 توضیح دهید (هر کدام در حداکثر سه خط) و جایگاه آن‌ها را با هم مقایسه کنید (۱۰ نمره)

۲. می‌خواهیم امکانی را ایجاد کنیم که بتوانیم یک متد استاتیک را از روی کامپیوتر دیگری از طریق شبکه فراخوانی کنیم. به این فرایند فراخوانی راه‌دور متد[1] گفته می‌شود. فرض می‌کنیم این امکان در جاوا وجود ندارد و ما می‌خواهیم آن را ایجاد کنیم. برای این کار باید اسم کلاس و متد موردنظر را به همراه پارامترهای متد برای کامپیوتر مقصد ارسال کنیم. روی کامپیوتر مقصد باید برنامه‌ای در حال اجرا باشد که این مقادیر را می‌گیرد، متد موردنظر را فراخوانی می‌کند و خروجی آن را از طریق شبکه برمی‌گرداند. تست زیر یک نمونه از اجرای این فرایند را نشان می‌دهد

```
@Test
public void testRPC() throws Exception{
  int port = 7890;
  RPC_Server server = new RPC_Server(port);
  server.listen();

  InvocationVO in = new InvocationVO("java.lang.Integer", "valueOf", "12");
  RPC_Client rpc = new RPC_Client("localhost", port);
  Object returnValue = rpc.invoke(in);
  Assert.assertEquals(returnValue, new Integer(12));

  server.shutdown();
}
```

کلاس RPC_Server روی کامپیوتر مقصد (فراخوانی کننده متد) اجرا می‌شود. کلاس RPC_Client روی کامپیوتر مبدأ (درخواست دهنده فراخوانی متد) استفاده می‌شود. در تست فوق، سرور و کلاینت هر دو روی یک کامپیوتر اجرا شده‌اند (localhost). یک شیء از کلاس InvocationVO اطلاعات مربوط به کلاس، متد و پارامترها را منتقل می‌کند.

```
public class InvocationVO implements Serializable{
  public String fullClassName;
  public String methodName;
  public Object[] parameters;
  public InvocationVO(String clazz, String method, Object... params) {
    this.fullClassName = clazz;
    this.methodName = method;
    this.parameters = params;
  }
}
```

[1] Remote Procedure Call (RPC) و یا Remote Method Invocation(RMI)

بخشی از کلاس RPC_Server در ادامه آمده است. بخش مشخص شده از این کلاس را پیاده‌سازی کنید. همچنین کلاس RPC_Client را به صورت کامل پیاده‌سازی کنید. (۲۵ نمره)

```
public class RPC_Server {
  private final class ThreadExtension extends Thread {
    private final ServerSocket server;
    private ThreadExtension(ServerSocket server) {
      this.server = server;
    }
    public void run() {
      while (alive) {
        try {
            هرچه لازم است برای این قسمت پیاده‌سازی کنید
        } catch (Exception e) {
          e.printStackTrace();
        }
      }
    }
  }
  private int listeningPort;
  private volatile boolean alive = true;
  public RPC_Server(int remotePort) {
    this.listeningPort = remotePort;
  }
  public void listen() throws IOException {
    final ServerSocket server = new ServerSocket(listeningPort);
    new ThreadExtension(server).start();
  }
  public void shutdown() {
    alive = false;
  }
}
```

۳. تابعی با نام ***createMap*** بنویسید که یک List به عنوان پارامتر بگیرد، تکرار هر یک از اعضای آن را بشمارد و در قالب یک Map برگرداند. کلید در این Map، اعضای List و مقدار آن تعداد تکرار هر عضو خواهد بود.(۱۰ نمره) خروجی این کد:

```
List<String> list1 = Arrays.asList("ali", "taghi", "ali");
List<Integer> list2 = Arrays.asList(1, 2, 3, 1, 2);
System.out.println(createMap(list1));
System.out.println(createMap(list2));
```

این گونه خواهد بود:

```
{ali=2, taghi=1}
{1=2, 2=2, 3=1}
```

۴. می‌خواهیم برای نگهداری یک صف از اشیاء، یک کلاس پیاده‌سازی کنیم که از لیست پیوندی (Linked List) یک‌طرفه برای نگهداری اشیاء استفاده می‌کند. با کمک متد enqueue یک شیء به انتهای صف اضافه می‌شود و با کمک متد dequeue یک شیء از ابتدای صف خارج و برگردانده می‌شود. کلاس موردنظر واسط (interface) QueueInt را پیاده‌سازی می‌کند و تست testStringQueue را پاس می‌کند. کلاس صف و هر چیز دیگری که لازم است را پیاده‌سازی کنید. پیاده‌سازی لیست پیوندی به عهده خودتان است و نباید از کلاس LinkedList یا کلاس‌های مشابه استفاده کنید. ارجاع به اولین و آخرین اعضای صف را نگهدارید تا عملیات ورود و خروج به صف با کارایی مناسب اجرا شود. (۱۵ نمره)

```
class EmptyQueueException extends Exception{}
.....................................................
interface QueueInt<T>{
	void enqueue(T t);
	T dequeue() throws EmptyQueueException;
}
.....................................................
@Test
public void testStringQueue() throws EmptyQueueException{
	Queue<String> queue = new Queue<>();
	queue.enqueue("me");
	queue.enqueue("you");
	Assert.assertEquals(queue.dequeue(), "me");
	Assert.assertEquals(queue.dequeue(), "you");
	try{
		queue.dequeue();
		Assert.fail();
	}catch(Exception e){
		// Do nothing
	}
```

یادآوری: در نوشتن تست‌های JUnit، تابع `assertEquals` اگر دو پارامترش با هم مساوی نباشند اعلام خطا می‌کند و `fail` همیشه اعلام خطا می‌کند (چیزی را چک نمی‌کند)

۵. برنامه‌ای شامل مفاهیم زیر بنویسید. هر یک از این مفاهیم ممکن است یک کلاس، شیء، واسط، کلاس مجرد، متد یا فراخوانی متد باشند. چیزی بیش از مفاهیم زیر لازم نیست پیاده سازی شود (۱۵ نمره)

«رونالدو» و «شفچنکو» دو فوتبالیست هستند. هر فوتبالیست یک انسان است. هر فوتبالیست یک «قیمت‌دار» است. هر انسان لزوماً یک «قیمت‌دار» نیست. هر انسان ویژگی «نام» دارد. هر «قیمت‌دار» ویژگی «قیمت» دارد. ساخته شدن یک نمونه شیء انسان بدون داشتن «نام» بی‌معنی است. ساخته شدن یک نمونه شیء فوتبالیست بدون داشتن «نام» بی‌معنی است. نوع یک شیء نمی‌تواند دقیقاً «انسان» باشد، ولی می‌تواند یک نوع دقیق‌تر مثل فوتبالیست یا دانشجو باشد. دانشجو یک «قیمت‌دار» نیست. «علی علوی» یک دانشجو است. ساخته شدن یک نمونه شیء دانشجو بدون داشتن «نام» و «شماره دانشجویی» بی‌معنی است. هر انسان می‌تواند طرفدار بعضی‌های دیگر باشد. علی علوی طرفدار رونالدو و شفچنکو است.

۶. خروجی برنامه زیر چیست؟ (۱۵ نمره)

```
interface Animal {
  void eat(Object o);
}
interface CanTalk {
  void talk(String s);
}
class Bird implements Animal {
  public Bird(){
    System.out.println("Constrcutor of Bird");
  }
  public void eat(Object o) {
    System.out.println("eat() in Bird");
  }
}
class Parrot extends Bird implements Animal, CanTalk {
  public void eat(String o) {
    System.out.println("eat() in Parrot");
  }

  public void talk(String s) {
    System.out.println("talk() in Parrot");
    if(s==null || s.length()>7)
      throw new BadStringException();
  }
}
class BadStringException extends RuntimeException {}

public class Zoo {
  public static void main(String[] args) {
    Bird bird = new Parrot();
    CanTalk cantalk = new Parrot();
    Object hard = new String("Shesh sikh jegar");
    Object easy = new String("jegar");
    try {
      bird.eat(hard);
      bird.eat((String)easy);
      cantalk.talk((String) easy);
      cantalk.talk((String) hard);
    } catch (BadStringException e) {
      System.out.println("BadStringException");
    } catch (Exception e) {
      System.out.println("Exception");
    } finally {
      System.out.println("The End... Good Bye!");
    }
  }
}
```

۷. پس از کش‌وقوس‌های فراوان، انجمن IOJME[2] به همراهی یونیسف، مدرس درس جاوا را مجبور کردند که نمرات درس را روی نمودار ببرد. چهار متد (curve1 ، curve2 ، curve3 و curve4) برای روی نمودار بردن آرایه نمرات نوشته شده است. کدام‌یک از این ۴ متد نمرات را تغییر نمی‌دهند؟ چرا؟ (یک توضیح کوتاه برای هر کدام) ممکن است هیچ کدام یا چند تا از متدها جواب سؤال باشند. تابع sqrt جذر پارامترش را برمی‌گرداند. انتخاب گزینه‌های اشتباه، نمره منفی دارد. (۱۰ نمره)

```
private static void curve1(double[] grades) {
  for (double d : grades) {
    d = Math.sqrt(d * 20);
  }
}
private static void curve2(double[] grades) {
  for (int i = 0; i < grades.length; i++) {
    curveGrade(grades[i]);
  }
}
private static void curve3(double[] grades) {
  double[] temp = grades;
  curve4(temp);
}
private static void curve4(double[] grades) {
  for (int i = 0; i < grades.length; i++) {
    grades[i] = Math.sqrt(grades[i] * 20);
  }
}
private static void curveGrade(double i) {
  i = Math.sqrt(i * 20);
}
```

۸. در زبان جاوا (و نه در JUnit) ، Assertion ها چه هستند و چه کاربردی دارند؟ برای نحوه استفاده از آن‌ها یک مثال بزنید. (۱۰ نمره اضافی)

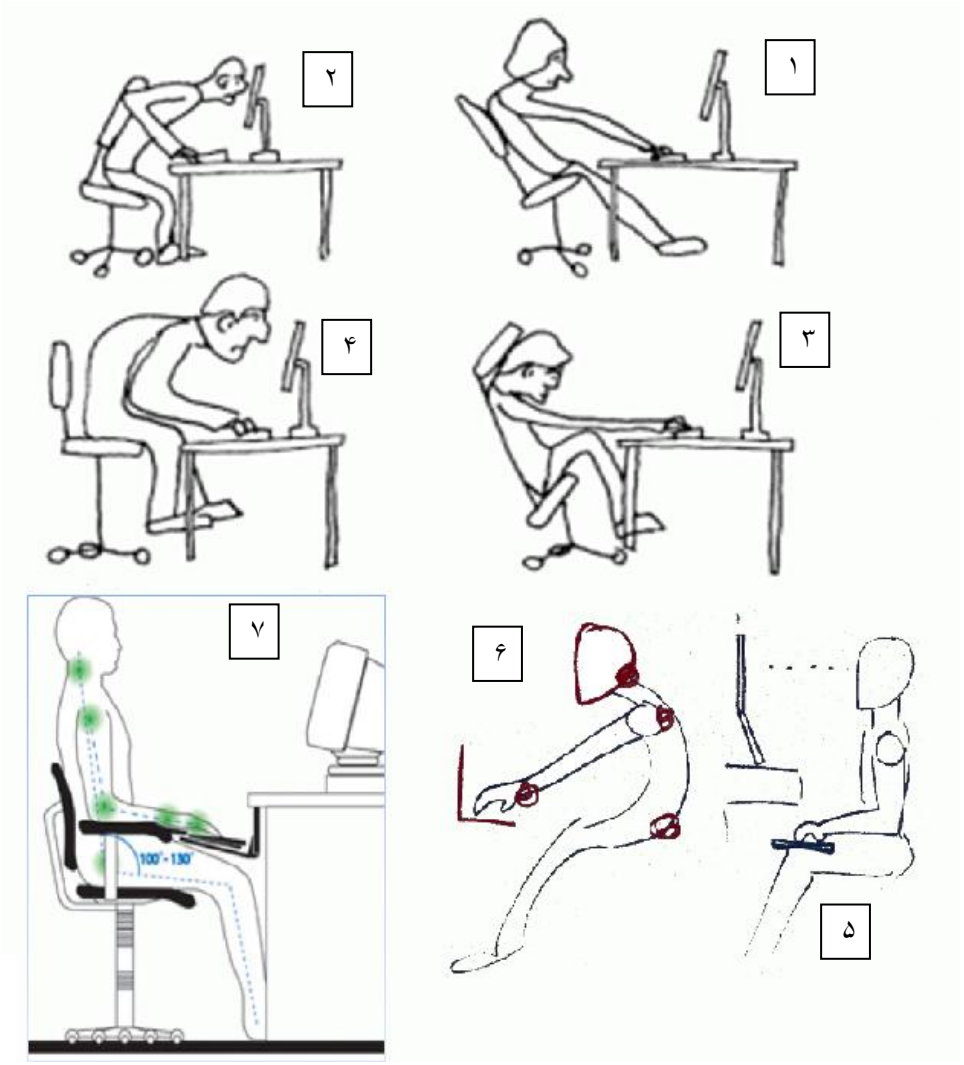
۹. هر یک از شکل‌های زیر نحوه نشستن یک برنامه‌نویس جاوا را نشان می‌دهد.

الف) تخمین بزنید هر یک چند سال می‌توانند به برنامه‌نویسی ادامه دهند.

ب) نحوه نشستن شما به کدام‌یک شبیه‌تر است؟

(۵ نمره اضافی)

[2] INTERGALACTIC ORGANIZATION for JUSTIFIED MARK EVALUATION



سلامتی مهمترین چیز است. مهمتر از ترقی، مهمتر از پول، مهمتر از قدرت... (هایمن راث، پدرخوانده)
(من با بخش دوم حرفش موافقم)

سلامت و سربلند و موفق باشید.
علی‌اکبری