202001 学期《Java 程序设计》线下考试答案及评分标准（B 卷）

一．卷选择题答案：

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
C	D	C	B	B	B	B	D	C	C	C	D	A	B	B	B	C	D	B	A

二．判断题答案：

1	2	3	4	5	6	7	8	9	10
√	×	√	√	√	×	×	√	×	×

三．完善程序填空题（每空 2 分）

1) x=_x; y=_y;
2) (leftBottom.x<=x)&&(x<=rightTop.x)　　&&(leftBottom.y<=y)&&(y<=rightTop.y)
3) System.arraycopy(c, 0, cords, 0, n);
4) if (cords[i].isInZone(leftBottom, rightTop)) return true;
5) continue;
6) Scanner(s.getInputStream());
7) PrintWriter(s.getOutputStream(),true);
8) break;
9) s = ss.accept();
10) serverThread.start();

四 、 程序设计题（每小题 10 分，共 30 分）

1、 给定一个 Listener 接口，它有如下方法：
1) void onEvent(int eventId): 接收一个事件，其编号为 eventId。

试编写一个 EventList 类，实现 Listener 接口:
1) void onEvent(int eventId): 收集所有收到的事件 eventId.　（2 分）
2) 并且实现方法 String toString()，将收集到的所有事件按先后次序，转换为字符串（以逗号分隔每个事件 id）。（3 分）

编写一个类 RandomEventGenerator，实现下述方法：
1) RandomEventGenerator(Listener listener): 存放 listener 到 this.listerner （1 分）

2) void generate(int stop)：每隔一秒，随机生成一个[0,100]内的整数 x，当 x 为奇数时，调用 Listener 接口方法 onEvent(x)，当 x=stop 时，返回。（2 分）

3) public static void main()：测试 RandomEventGenerator.generate 和 EventList.toString 两个方法是否正常工作。 （2 分）

```
import java.util.*;
interface Listener {
     void onEvent(int eventId);
}
class EventList implements Listener {
     private ArrayList<Integer> lst = new ArrayList<>();
     public void onEvent(int eventId) { lst.add(eventId);  }          //2 points
     public String toString() {
          int size = lst.size();
          if (size == 0) return "";
          StringBuffer b = new StringBuffer();
          for(int i=0; i<(size-1); ++i) {
               b.append(lst.get(i));                                  //3 points
               b.append(",");
          }
          b.append(lst.get(size-1));
          return b.toString();
     }
}
public class RandomEventGenerator {
     private Listener listener;
     public RandomEventGenerator(Listener listener) {
          this.listener = listener;                                  //1 points
     }
     public void generate(int stop) {
          Random r = new Random();
          int x;
          do {
               x = r.nextInt() % 100;
               if (x < 0) x = -x;
               if (x % 2 == 1)
                    listener.onEvent(x);                             //2 points
          }
          while(x != stop);
     }
     public String toString() { return listener.toString(); }
     public static void main(String[] args) {
          RandomEventGenerator g = new RandomEventGenerator(new EventList());
          g.generate(11);
          System.out.println(g);                                     //2 points
     }
}
```

2、 试编写一个设备类 Equipment，它有下述属性：

1）String name：设备名称

2）int num：该设备的数量

3）float price：该设备的单价

再编写一个 EquipmentCollection 类，实现下述方法：

1） void add(Equipment e)：增加一个设备

2） List<Equipment> sortByTotalPrice()：以总价（num * price）由小到大的顺序排序所有增加的设备，并返回列表。

3） public static void main(String[] args)：随机生成一组设备，并依次增加到 EquipmentCollection 实例中，测试 sortByTotalPrice 是否正常工作。

```
import java.util.*;
class Equipment {
     public String name;
     public int num;
     public float price;
     public Equipment(String name, int num, float price) {
          this.name = name;
          this.num = num;
          this.price = price;                                          //1 points
     }
     public String toString() {
          return name + "t_" + num * price + "__" + num + "_" + price;
     }
}
public class EquipmentCollection {
     private ArrayList<Equipment> arrs = new ArrayList<>();
     public void add(Equipment e) { arrs.add(e); }                     //1 points
     public List<Equipment> sortByTotalPrice() {
          arrs.sort(new Comparator<Equipment>() {
               public int compare(Equipment a, Equipment b) {
                    float sp = a.num * a.price;
                    float sp2 = b.num * b.price;
                    if (sp > sp2) return 1;
                    if (sp < sp2) return -1;                           //4 points
                    return 0;
               }
          });
          return arrs;
     }
     public String toString() {
          StringBuffer b = new StringBuffer();
          for(Equipment e: arrs) { b.append(e.toString() + "\n"); }
          return b.toString();
     }
     public static void main(String[] args) {
          EquipmentCollection c = new EquipmentCollection();
          Random r = new Random();
          int num; float price;
          for(int i=0; i<10; ++i) {
               num = r.nextInt() % 10;
               if (num < 0) num = -num;
               price = r.nextFloat() * 100;
               c.add(new Equipment("" +i, num , price));              //2 points
          }
          System.out.println(c);
          List<Equipment> lst = c.sortByTotalPrice();
          for(Equipment e: lst) { System.out.println(e); }            //2 points
          System.out.println();
     }
}
```

3、 实现一个 Master 类和 Slaver 线程类，以缓冲区通信方式对每一批次的整数求和，具体如下：

Master 类实现下述方法：

1) Master():创建一个线程类 Slaver 实例 self.slaver。

2) loop(int low, int high, int num, int seconds):

2.1) 每隔 seconds 秒，随机生成一个批次的整数集（共 num 个），每个整数在[low, high]区间，并将每一批次的整数集发送给 Slaver 实例 self.slaver 以便求和.

2.2) 若 Master 收到 self.slaver 对某一批次的整数集的求和结果，则打印输出结果，若没有收到，则继续执行 2.1)。 （6 分）

Slaver 线程类在没有收到一个批次的整数集时，就选择睡眠 1 秒钟，若收到一个批次的整数集时，则求和，并将结果发送回 Master。 （4 分）

要求：Master 类和 Slaver 类完成线程同步以便保证不能漏掉任何一批次的整数集及其求和结果。

```
import java.util.*;
class Piple {
     private LinkedList<ArrayList<Integer>> inputs = new LinkedList<>();
     private LinkedList<Integer> results = new LinkedList<>();
     synchronized public ArrayList<Integer> pollData() {
          return inputs.pollFirst();                                        //1 points
     }
     synchronized public void addData(ArrayList<Integer> data) {
          inputs.add(data);                                                 //1 points
     }
     synchronized public Integer pollResult() {
          return results.pollFirst();                                       //1 points
     }
     synchronized public void addResult(Integer r) {
          results.add(r);                                                   //1 points
     }
}
class Slaver extends Thread{
     private Piple pip;
     public Slaver(Piple pip) {this.pip = pip; }
     public void run() {
          while(true) {
               ArrayList<Integer> data = pip.pollData();
               if (data == null) {
                    try { Thread.sleep(1000); }
                    catch(Exception e) {System.out.println(e); }
                    continue;
               }
               int sum = 0;
               for(Integer e: data) { sum += e; }                           //2 points
               pip.addResult(sum);
          }
     }
}
public class Master {
     private Slaver s;
     private Piple pip;
     private Random r = new Random();
     public Master() {
          pip = new Piple();
          s = new Slaver(pip);
     }
     private ArrayList<Integer> getBatch(int low, int high, int num) {
          ArrayList<Integer> res = new ArrayList<>();
          float gap;
          for(int i=0; i<num; ++i) {
               gap = r.nextFloat();
               gap = gap * (high - low);                                    //1 points
               int v = (int)gap + low;
               if (v > high) v = high;
               res.add(v);
          }
          return res;
     }
     public void loop(int low, int high, int num, int seconds) {
          s.start();                                                        //1 points
          while(true) {
               try { Thread.sleep(seconds); }
               catch(Exception e) {System.out.println(e); }
               ArrayList<Integer> batch = getBatch(low, high, num);
               pip.addData(batch);                                          //1 points
               for(Integer d: batch) { System.out.print(d + ", "); }
               System.out.println();
               Integer res = pip.pollResult();                              //1 points
               if (res == null) continue;
               System.out.println("res: " + res);
          }
     }
     public static void main(String[] args) {
          Master m = new Master();
          m.loop(1, 10, 5, 5);
     }
}
```

五 、 附加题（30 分）

参考答案：
解法 1：主要逻辑是利用 Map 和递归函数对输入进行预处理(10 分)，注意对反斜杆的处理需要与查询、输出的处理统一（5 分），查询方面主要是对 Map 的使用（5 分）并使用递归的方式处理好“.”连接符号（5 分）和输出（5 分）。

解法 2：不对输入进行过多的预处理，主要是将输入拼成一个长字符串（5 分），然后针对查询，利用子串查询（或正则匹配）获取到相应部分内容（10 分），再使用递归或递推处理好“.”连接符（5 分），同时要注意对限定好连接符后面的子串查询区域（5 分），并做好输出即可（5 分）。

出现拼写、语法差错可以酌情扣分。
其他解法，可根据主要逻辑思路正确与否酌情给分。