主管领导审核签字

哈尔滨工业大学 2019 学年春季学期

软件构造　　试　题

题号	1	2	3	4	5	6	7	总分
得分								
阅卷人								

片纸鉴心　诚信不败

授课教师＿＿＿＿　密　姓名＿＿＿＿　封　学号＿＿＿＿　线　院系＿＿＿＿

本试卷满分 100 分，折合 60%计入总成绩。

1　单项选择题（每个 2 分，共 30 分。请将全部答案填写至下表中，否则视为无效）

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
(9)	(10)	(11)	(12)	(13)	(14)	(15)	

(1)　以下___中的软件构造实体，前者属于 build-time view，后者属于 moment-view

A　Software Configuration Item (SCI)　　Stack Trace
B　Code Snapshot　　Test Case
C　Static linking library　　Memory Dump
D　Abstract Syntax Tree (AST)　　Code Churn

(2)　以下工具/命令___无需执行正在开发的软件代码即可获得期望结果

A　JUnit test 和 EclEmma
B　Checkstyle 和 git commit
C　Eclipse Memory Analyzer (MAT)与 jmap
D　SpotBugs 和 VisualVM

(3)　关于 immutability 和 mutability 的说法，最恰当的是____

A　被 static 和 final 修饰的 Date departTime 是 immutable 的
B　采用 State 设计模式以使对象具备状态转换功能的 ADT 是 mutable 的
C　Collections.*unmodifiableList*(new ArrayList<String>())得到的 List 对象是 immutable 的，程序运行期间其内容无法变化
D　new ArrayList<String>().iterator()得到的结果是 immutable 的

(4)　关于 ADT 的 Rep、AF、RI 的说法，正确的是___

A　Immutable 类的对象，其 rep 自对象创建之后就不能再发生变化
B　Rep exposure 仅针对 immutable 的 ADT 来说有意义，一旦发生 rep 泄露可能导致 RI 被违反；对 mutable 的 ADT，由于其 rep 本来就可变化，故无需考虑 rep exposure

C AF 作为一种关系，具备的性质是：满射、双射，但并非总是单射

D RI 可看作一组条件约束，rep 满足这些条件的对象才是合法的

(5) 以下关于方法 spec 的说法，不恰当的是___

A 若客户端传递进来的参数不满足前置条件，则方法可直接退出或随意返回一个结果

B 方法的 spec 描述里不能使用内部代码中的局部变量或该方法所在类的 private 属性

C 方法 A 的前置条件比方法 B 的前置条件更强，后置条件相同，那么开发者实现 A 的难度要高于实现 B 的难度

D 如果修改了某个方法的 spec 使之强度变弱了，那么 client 调用该方法的代价可能变大了，即 client 需要对调用时传入该方法的参数做更多的检查

(6) 某方法 public Number info(Set<Number> a) throws Exception，以下__是它的合法 override

A `private Object info(Set<Number> m) throws Exception`

B `public Number info(HashSet<? extends Number> m)`

C `public Number info(Set<Integer> m, Number n) throws IOException`

D `public Integer info(Set<Number> m) throws IOException, InterruptedException`

(7) 关于 ADT 的 equals 和 hashCode 的说法，正确的是____

A Immutable 的 ADT 对象 a 和 b，若 a.hashCode()=b.hashCode()，那么 a.equals(b)一定为真

B Override equals()的时候，equals()的代码中需要逐个比较 rep 中每一个域的值是否相等

C 某 ADT 的对象 a 和 b,若 a.equals(b)为假,那么该 ADT 不应存在任何方法 op 使得 a.op()=b.op()

D 已知 List 及其子类型实现的是观察等价性，那么针对以下代码中的 a 和 b，a.equals(b)为真

```
List<String> a = Arrays.asList(new String("c"));
List<String> b = Arrays.asList("c");
```

(8) 针对 assertion 和 exception 的说法，最恰当的是___

A 代码中的 assert 语句太多，会影响代码运行性能，这是为了 correctness 所需付出的必要代价

B 代码执行期间抛出的 RuntimeException 及其子类型异常是程序员在代码中无法捕获的，而所有抛出的 checked 异常均需利用 try 和 catch 进行捕获和处理

C 若为了代码的简洁性更高，开发者应更倾向于使用 unchecked 异常；若为了程序的健壮性更高，则应倾向于使用 checked 异常

D 某 public 方法的第一行是 if(! pre-condition) {...}，那么用以下每行代码分别替换“...”，在任何情况下所起的效果是一样的：

```
throw new AssertionError();
throw new IllegalArgumentException();
assert false;
```

(9) 某程序运行期间的控制台输出如下所示，以下说法最恰当的是____

```
Exception in thread "main" exception.CarAlreadyParkingException:
                    MN007 is already in parking field, cannot be parked again
    at field.A.parking(A.java:111)
    at field.B.parking(B.java:40)
    at field.C.parking(C.java:44)
    at application.D.main(D.java:55)
```

A 该异常发生的第一现场是 D 类的 55 行，在 A 类的第 111 行被捕获

B 不确定此次抛出的是一个 unchecked 类型还是 checked 类型的异常

C A、B、C 类均有名为 parking 的方法，且其 spec 中一定包含 throws CarAlreadyParkingException

D 该异常未被捕获，也未被处理

(10) 关于测试的说法，正确的是___

A 所用测试用例的数量越多，代表对代码的覆盖度越高，发现 bug 的能力就越强

B 如果某方法返回值类型是 void，则无法对其单独进行测试，必须联合 ADT 的其他方法进行联合测试

C 如果某 bug 已被正确修复并已通过测试，那么为了降低后续测试的代价，应将该 bug 对应的测试用例从测试库中删除

D 对只有 getXXX()和 setXXX(...)方法的 ADT，因为其业务逻辑非常简单，无需为其撰写测试用例

(11) 某个 ADT 的 rep 中有一个 static int 类型的变量 a、有一个 final String 类型的变量 b，在该 ADT 的某方法中使用了一个类型为 int[]的局部变量 c。那么在 Java 8 环境下运行程序时，变量 a、b、c 所指向的存储空间分别在____

A	Metaspace	heap	heap
B	Heap	metaspace	stack
C	Heap	perm	heap
D	Perm	metaspace	stack

密

(12) Java 程序在启动时，通过 JVM 参数配置不能做到的是___

A 设置运行中使用的 GC 算法

B 设置运行中可占用的 heap 最大尺寸、young generation 和 metaspace 的最大尺寸

C 设置运行过程中两次 GC 的最大时间间隔

D 设置将 GC 日志写入文件的路径

(13) 针对 ADT 的性能，以下说法不恰当的是____

A 防御式拷贝避免了 ADT 表示泄露，但增加了程序运行时内存消耗和 GC 的时间代价

封 B 带泛型的 ADT，在运行时需要进行动态的类型匹配，会造成程序运行时间的轻微增加

C 使用 JConsole 或 VisualVM 进行程序性能监控和分析，会造成程序运行性能的轻微下降

D 使用 Singleton 或 Flyweight 模式设计 ADT 的目的是降低因 new 对象和频繁 GC 而带来的性能损失

(14) 关于多线程 Java 程序及其 threadsafe，以下说法正确的是___

A 线程执行 Thread.*sleep*(1000)期间，该线程休眠之前所获得的锁会被释放给其他线程

B 即使是通过 Collections.*synchronizedList*(...)得到的 List 对象，也难以在任何多线程场景下做到 threadsafe

C 若 ADT 的 rep 所有属性是 final 和 immutable 的，则它在任何多线程场景下都可做到 threadsafe

D 操作系统和 JVM 在调度多线程 Java 程序执行时，以单个 Java 方法或操作符为基本单元进行 interleaving

线

(15) 以下代码执行结束后，用下图描述内存里的对象状态（假设代码执行期间 JVM 未进行任何 GC）。关于该图中存在的错误及其修改，说法正确的是___

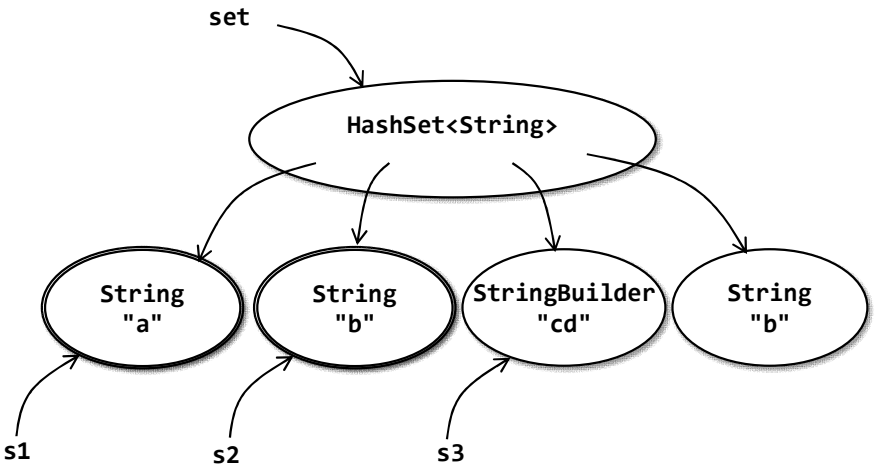
```
String s1 = new String("a");
String s2 = "b";
final StringBuilder s3 = new StringBuilder("c");
final Set<String> set = new HashSet<>();
set.add(s1);
set.add(s2);
set.add(s3.toString());
s3.append("d");
set.add("b");
```

A 因为 s3 由 final 所修饰，s3 所指向的椭圆应为双线椭圆，且连接二者之间的箭头应为双线箭头

B 最右侧的 String("b")以及 HashSet<String>指向它的箭头不应该存在

C HashSet<String>不应有箭头指向 StringBuilder("cd")，而应有箭头指向一个新的双线椭圆 String("cd")

D 因为 set 是被 final 修饰的，故从 HashSet<String>向外指出的四个箭头均应为双线箭头



——揉揉脑袋，准备进入后半程——

密

封

线

以下为分析与设计题，共 70 分。

某公司拟设计和开发一个停车场管理系统，其基本需求陈述如下：

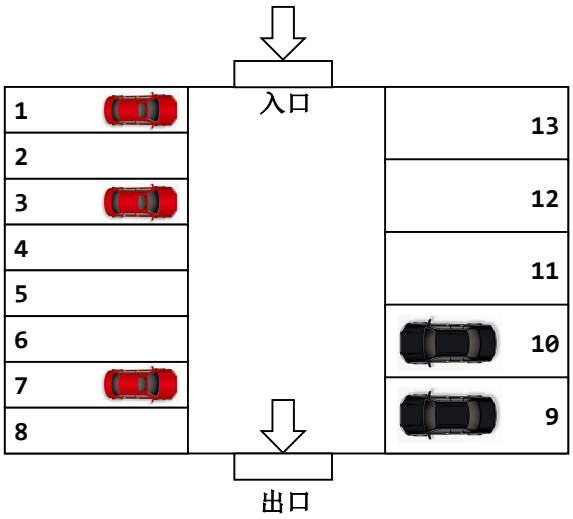
(1) 一个停车场有 n 个车位(n>=5)，不同停车场包含的车位数目不同。

(2) 一辆车进入停车场，如果该停车场有空车位且其宽度足以容纳车的宽度，则可以在此停车。

(3) 停在停车场里的车，随时可以驶离停车场，根据车型或时间自动计费。

(4) 停车场管理员可以随时查看停车场的当前占用情况。

下图给出了一个包含 13 个停车位的小型停车场示例图，其中 1-8 号停车位较窄，9-13 号停车位较宽。在当前状态下，第 1、3、7、9、10 号车位被占用，其他车位空闲。



用于刻画“停车场”的 ADT 被命名为 ParkingField，这是一个接口，其**完整代码**如下所示：

```
/**
 * A mutable ADT
 */
public interface ParkingField {

    /**
     * 在某个停车位上停车
     * @param    c                          要停进来的车辆，not null
     * @param    num                        指定的停车位编号，自然数
     * @throws   LotOccupiedException       如果该停车位已被其他车辆占用
     * @throws   NoSuchLotException         如果该停车场没有该编号的停车位
     * @throws   LotTooNarrowException      如果该停车位的宽度小于该车辆的宽度
     * @throws   CarAlreadyParkingException 如果该车已经停在该停车场
     */
    public void parking (Car c, int num)  throws LotOccupiedException, NoSuchLotException,
                                          LotTooNarrowException, CarAlreadyParkingException;

    /**
     * 在停车场停车，自动分配空闲停车位
     * @param    c                          要停进来的车辆，not null
     * @throws   ParkingFieldFullException  若该停车场已满（不存在空闲停车位）；或有空闲停车位，但
     *                                      它们的宽度均小于该车辆的宽度
     * @throws   CarAlreadyParkingException 如果该车当前已经停在该停车场
     */
    public void parking (Car c) throws ParkingFieldFullException, CarAlreadyParkingException;

    /**
     * 将汽车驶离停车场
     * @param    c                          待驶离的车辆，not null
     * @return                              本次停车的费用
     * @throws CarNotInFieldException       如果该车辆当前并未停在该停车场内
     */
    public double departure (Car c) throws CarNotInFieldException;

    /**
```

```
     * 返回当前停车场的状态
     * @return   Key 为停车位的编号，Value 为该车位上的车辆。如果停车位上无车辆，则对应的 Value 为 null
     */
    public Map<Integer, Car> status ();

    /**
     * 当前停车场是否已满
     * @return                              true：已满；false：尚有空闲停车位
     */
    public boolean isFull ();

    /**
     * 为车辆分配该停车场的某个空闲停车位
     * @param    c                          要停进来的车辆，not null
     * @return                              一个空闲停车位的编号，且其宽度大于 c 的宽度
     * @throws ParkingFieldFullException    如果该停车场已经满了 (不存在空闲停车位)
     *                                      或者有空闲停车位，但它们的宽度均小于该车辆的宽度
     */
    public int getOneFreeLot (Car c) throws ParkingFieldFullException;
}
```

该接口 ParkingField 的某个具体实现类为 ConcreteParkingField，其代码如下：

```
//A mutable class 一个停车场
public class ConcreteParkingField implements ParkingField {
    private final String name;        //停车场名称，不能为空，长度大于 0
    private final int lotsNumber;     //车位数量，最少 5 个车位

    //停车场包含的停车位对象的集合，集合中元素的类型是 Lot，不同停车位对象的编号不能相同
    private final Set<Lot> lots = new HashSet<>();

    //停车场当前状态，Key 为一个 Lot 对象，Value 为当前停在该 Lot 对象所代表的车位上的 Car 对象
    //且 Key 和 Value 的值均不能为 null，意即：如果一个停车位上没有停车，则它不应包含在该 map 中
    private final Map<Lot, Car> status = new HashMap<>();

    //以下为方法区域，但省略了构造函数和上述接口中各方法的具体实现

    /**
     * 根据停车位号码获取停车位对象
     * @param    num                        指定的停车位编号，自然数
     * @return                              num 所代表的停车位的 Lot 对象
     * @throws   NoSuchLotException         如果该停车场没有该编号的停车位
     */
    private Lot getLotByNumber(int num) throws NoSuchLotException {...}
}
```

其他辅助 ADT 的代码如下：

```
//An immutable class 一辆车
public class Car {
    private final String plateNo;                   //车牌号
    private final int width;                        //车辆宽度，自然数 (度量单位为厘米)，>100

    public Car(String plateNo, int width) {...}     //构造器
    public String getPlateNo() {...}                //返回车牌号
    public int getWidth() {...}                     //返回宽度

    @Override
    public boolean equals(Object car){...}
    @Override
    public int hashCode() {...}
}
```

```
//An immutable class 一个停车位
public class Lot {
    private int number;                         //停车位编号，自然数
    private int width;                          //停车位宽度，自然数 (度量单位为厘米)，>150
```

```
    public Lot(int number, int width) {...}    //构造器
    public int getLotNumber() {...}            //返回停车位的自然数编号
    public int getWidth() {...}                //返回宽度
    public boolean wideEnough(Car c) {...}     //判断该停车位宽度是否足够停入车辆
                                               //true表示可以，false表示不可以
    @Override
    public boolean equals(Object lot) {...}
    @Override
    public int hashCode() {...}
}
```

注意：在以下所有题目中，除非明确说明，否则不能为上述 ADT 添加任何其他方法或 rep、不允许定义新的 ADT 和异常类。各题目的要求是独立的，做某题目的时候无需考虑其他题目中给出的新设定。

2 (10 分) 请为 ParkingField 接口增加一个静态工厂方法，构造并返回一个 ConcreteParkingField 对象。请给出静态工厂方法的 spec 和 Java 代码。由于 ConcreteParkingField 类代码中尚未给出其构造函数，你需要根据其 rep 设计和实现其合理的构造函数，以便将其用于你的静态工厂方法中。

```
/** 在此区域撰写静态工厂方法的 spec，不允许在参数中显式使用 Lot 类型 (3 分)
 *
 *
 *
 *
 *
 *
*/
```

```
//在此区域撰写该方法的 signature (返回值、方法名、参数列表、异常等)和实现代码 (3 分)
public
{









}
```

```
//在此区域写出 ConcreteParkingField 的构造函数，不允许在参数中显式使用 Lot 类型 (4 分)
public
{









}
```

密 封 线

3 (15 分) 请写出 ConcreteParkingField 类的 RI 和 checkRep()。

```
//在此区域撰写 RI，使用中文或英文均可 (7 分)
//
//
//
//
//
//
//
//
//
//
//
//
//
//
//
//
//
//
```

```
//在此区域撰写 checkRep() (8 分)
private void checkRep() {









}
```

本页的两个问题，二选一作答。若两个均作答，以第一个为准
可对已有 ADT 的 rep 和方法进行修改/扩展，可用文字、UML 图、伪代码等方式加以说明

4-1 (10 分) 有些停车场是无人管理的公共停车场，有些则由专门公司管理。上述 ConcreteParkingField 的 rep 不支持后者。请使用 decorator 设计模式，在不改变 ConcreteParkingField 的情况下，同时做到：(a) 在创建 ParkingField 对象时包含“公司”信息 (String company)；(b) 车辆在此类停车场进行停车(调用 parking 方法)和驶离(调用 departure 方法)时，能打印欢迎和告别信息。

(1) 简要阐述如何修改和扩展现有设计以实现该目标 **(7 分)**；

(2) 在 client 代码中，给定一个 ParkingField 对象，如何创建一个由“HIT”管理的停车场对象**(3 分)**。

```
ParkingField pf = ...;                //此处无需补充，假设 pf 已成功创建
ParkingField pfWithCompany = ...;     //用你的 decorator 方案，如何实现？
```

4-2 (10 分) 考虑将来对 ParkingField 的功能扩展，使用 visitor 模式改造当前设计。例如要扩展的一个功能是统计停车场当前时刻占用比例(=已停车的车位数量÷总车位数)。

(1) 简要阐述如何修改和扩展现有设计以实现该目标 **(7 分)**；

(2) 在客户端代码中，给定一个 ParkingField 对象，如何使用 visitor 统计当前时刻占用比例 **(3 分)**。

```
ParkingField pf = ...;    //此处无需补充，假设 pf 已成功创建
                          //并且进行了一系列 parking 和 departure 操作
double fullRatio = ...;   //用你的 visitor 方案，如何实现？
```

密

封

线

本页的两个问题，二选一作答。若两个均作答，以第一个为准

5-2 题目可对已有 ADT 的 rep 和方法进行修改/扩展

5-1 (10 分) 停车场管理系统启动时，主程序读入外部文本文件已创建一系列 ParkingField 对象。该文本文件遵循特定的语法格式，每个以 PF 开头的行代表一个 ParkingField 对象，语法说明如下所示。请写出：(1) PF 的产生式形式的语法定义 (5 分)；(2) PF 的正则表达式 (5 分)。

(1) PF::= 一个由<>括起来的字符串，分为三部分，分别代表停车场名字、最大车位数、公司名字，三部分之间由逗号“,”分割。

(2) 停车场名字 ::= 字符串，长度不限。可以由一个单词或多个单词构成，单词由字母或数字构成，单词之间只能用一个空格分开。

(3) 停车场最大车位数 ::= 自然数，其值最小为 5。不能为 012、0012 的形式，只能为 12 的形式。

(4) 公司名字 ::= 与停车场名字的语法规则一致，但可以为空。若该部分为空，表示该停车场没有公司管理 (即公共停车场)。

以下是三个例子:

```
PF::=<92 West Dazhi St,120,HIT>
PF::=<Expo820Roadside,10,>  //无公司管理的停车场，最后一个逗号之后为空
PF::=<73 Yellow River Rd,50,Harbin Institute of Technology>
```

5-2 (10 分) ParkingField 需要具备遍历其中所停的所有 Car 对象的能力。拟使用以下形式的 client 端代码，按车辆所在停车位编号由小到大的次序，逐个读取所停车辆。请扩展现有设计方案(修改/扩展哪些 ADT)，在下方给出你的设计思路描述，必要时可给出关键代码示例或 UML 类图辅助说明。

```
Iterator<Car> iterator = pf.iterator();  //pf 是一个类型为 ParkingField 的对象
while (iterator.hasNext()) {
    Car c = iterator.next();
    System.out.println("A car " + c + " is now parked in " + pf);
}
```

密 封 线

6 (10 分) ParkingField 接口中的 void parking (Car c, int num)方法的 spec 中给出了四个异常，如下所示，每个异常后面的文字表示当异常 e 抛出的时候用 e.getMessage()所能输出的信息：

LotOccupiedException	编号为 num 的车位已被其他车辆所占用
LotTooNarrowException	编号为 num 的车位宽度小于 c 的宽度，c 无法在此停车
NoSuchLotException	该停车场不存在编号为 num 的车位
CarAlreadyParkingException	车辆 c 已停在该停车场且没有驶离，不能再次停车。

假设上述四个异常均已经实现为 checked 异常，且均具有一个构造方法，其参数如下：

LotOccupiedException 和 NoSuchLotException	参数均为 int num
LotTooNarrowException	参数为(Car c, int num)
CarAlreadyParkingException	参数为 Car c

请利用之前的 ADT 代码与 spec 说明，写出该方法的 Java 实现代码。答题区域很大，无需全部填满。

```
public void parking(Car c, int num) throws LotOccupiedException,     NoSuchLotException,
                                           LotTooNarrowException,    CarAlreadyParkingException
{









}
```

7 (15 分) 为司机开发了一个客户端，允许司机寻找车位进行停车。为每个车辆设定一个线程，代码如下：

```
1  public class ParkingThread implements Thread {
2    private final ParkingField pf;    //司机要进入的停车场
3    private final Car car;            //司机驾驶的车辆
4    public ParkingThread(ParkingField pf, Car car)
5    {this.pf = pf; this.car = car; }
6    @Override
7    public void run() {
8       int freeLot = -1;
9       while (freeLot == -1) {
10          try {
11              freeLot = pf.getOneFreeLot(car);
12          } catch (ParkingFieldFullException e) {
13              Thread.sleep(1000);
14              continue;
15          }
16      }
17      pf.parking(car, freeLot);
18   }
19 }
```

(1) (6 分) 检查上述 19 行代码，找出无法通过 static type checking 之处，并指出如何修改正确。

(2) (9 分) 假设你已完成(1)的修改，之后在主程序中启动了多个上述线程，代码如下。为保证 thread safe，请指出如何修改 ConcreteParkingField、Lot、Car、ParkingThread 的代码，**以消除线程不安全风险，并能尽可能保证程序的并发执行效率。修改代码时不应改变程序已有逻辑。**

```
ParkingField pf = ...;      //假设 pf 已成功创建
List<Car> cars = ...;       //假设 cars 已成功创建
for(Car car : cars) {
    Thread t = new Thread(new ParkingThread(pf, car));
    t.start();
}
```