

科廷大学-计算机学科

**面向对象软件工程(COMP2003)**

学期2020

**实得考试**

**重量:产品唛头的35%。**

**的问题。**这篇试题包含四个(4)题。全部回答。总共70分。分配给每个问题的分数显示在问题的旁边。

**回答的格式。**将你所有的答案输入一个单词(.docx)、LibreOffice (.odt)或文本(.txt)文件(你的“答案文档”)。当你写答案时，清楚地指出相关的问题编号/字母。在开始时包括你的姓名和学生证。

**时间表。**从2020年6月16日(周二)上午8:30到6月23日(周二)上午8:30 (UTC+8)，你有7天(168小时)时间完成并提交答案。在此期间，你可以按照自己的意愿安排工作时间。然而，迟交的作品将不被接受。

**宣言》的创意。**你还必须完成并提交一份原创声明(无论是扫描的，拍照的，还是电子填写的)。空白表格可以在与此试卷相同的地方找到。

**提交。**提交(1)你的回答文件，(2)你的原创性声明。请将两篇论文提交到黑板上的“课后考试”区域。在截止日期之前，你可以提交任何次数。只有你的期末提交才会被评分。

您必须验证您的提交工作(通过重新下载您的提交)。正确提交完全是你的责任。

**参考资料。**这是一个公开的评估。你可以参考任何书面材料，包括你的笔记、课程材料、书籍、网站等。然而:

•您必须完全独立完成此评估。

•在评估期间，您不能与任何其他学生就考试进行交流。

•在评估过程中，你不能为了寻求或接受评估的任何部分的答案而与任何其他人交流。

•你的答案将通过文本匹配软件检查是否有作弊、串通和/或抄袭的迹象。

•问题的设计使得任何两个独立作业的学生都不能给出相同的答案。

**第1页共5页**

**灰尘实得考试**

**学期2020**

**问题1(20分)**

这个问题与观察者模式有关。

考虑一个游戏(2D或3D)，玩家控制一个角色在世界中移动，收集物品和可能的战斗生物。(这可以像《马里奥兄弟》那样简单，也可以像现代第一人称射击游戏那样复杂。)

游戏必须处理的事件之一是游戏世界中两个物体之间的物理碰撞。也就是说，一件东西撞上或掉到另一件东西上。碰撞的结果取决于碰撞的物体;例如，如果角色与某种“奖励”发生冲突，角色的生命值或财产将相应地被修改，而该奖励将消失。系统的一部分必须负责检测这种碰撞，另一部分(或多个部分)必须负责处理发生的事情。

还有其他各种各样的事件，包括字符到达一个特定的位置或使用一个特定的项目，一个特定的键盘/鼠标被按下，一个倒计时结束，等等)。对于每个事件，必须有东西产生它，也必须有东西处理它(比如，通过修改角色的状态，让角色做一些事情，修改敌人，或者修改世界本身)。

您的任务是使用Observer模式实现此类事件处理。必须处理的事件类型之一是两个游戏世界对象之间的冲突(如上所述)。

然而，你还需要选择:

•游戏中另外两种不同类型的事件(除了物体碰撞)。

•游戏中可能导致事件发生的两个不同的类。(结合起来，这两个类应该负责引起所有三种事件类型。)

•游戏中的两个不同的类会对这些事件做出反应。(结合起来，这两个类应该对所有三种事件类型做出反应。)

假设所有事件处理类都需要知道事件发生时的细节。

(你可以想象这是一种什么样的游戏，以及它的结构。)

要回答这个问题，请为每个类/接口编写代码。您可以使用Java、c++、Python或c#，但是要避免使用duck类型。没有必要提供UML。

显示每个类/接口中的所有相关字段和方法。您不需要在实际的事件处理代码中确切地显示发生了什么，但是要以注释的形式简要地描述它。

**问题2出现在下一页**

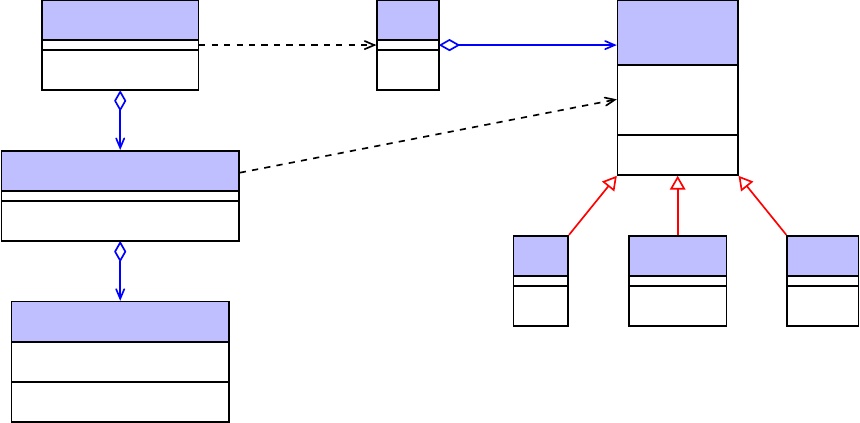
**第2页共5页**

**灰尘实得考试**

**学期2020**

**问题2(20分)**

这个问题涉及依赖注入、工厂和错误处理。假设你有以下的类结构:



GameLoader

+ load():游戏1解析器

+ parse():元素(1 . .\*] 1 Lexer -文件名:String + nextToken(): String

“创建”

游戏……

1 . . \*

元素- x: int - y: int…

砖

字符

敌人

…

…

…

这是初步的UML。为了回答这个问题，您将需要引入至少一个额外的类，并重构一些关系。

事情是这样的:

•GameLoader.load()负责从文件中加载保存的游戏。

•一个游戏对象有几个元素对象，每个对象代表游戏的一个元素。有三种类型，分别是Brick、Character和Enemy。

•输入的文件格式包含单词“砖块”，“字符”或“敌人”，然后是一个值为x，然后是一个值为y。例如:



角色5 10敌人11 1敌人10 1砖0 0砖3 9敌人5 3

•Lexer.nextToken()直接读取文件，或者下一个单词或数字，并以字符串形式返回。当它到达末尾时返回null。

•parse()获取标记并相应地创建一个元素对象列表。

•Lexer和解析器可能会遇到错误。如果是，则必须抛出GameLoadException。这组类中不需要捕捉那个异常。

您的任务是使用依赖注入实现GameLoader和Parser类。为此，您还需要创建一个工厂(或多个工厂)。上面显示的类本身不应该被视为工厂。不过，您可以正常创建GameLoadException对象，而不需要工厂或依赖项注入。

为GameLoader和Parser以及factory(ies)(但不包括其他类)提供完整的代码。对其他类的设计做出合理的假设。你可以使用Java, c++， Python或者c#。

**第三个问题出现在下一页**

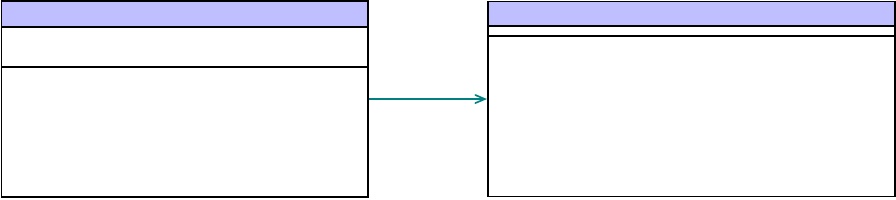
**第3页共5页**

**灰尘实得考试**

**学期2020**

**问题3(20分)**

这个问题涉及到国家模式。考虑以下类:



工具集-画布:画布

帆布

+ draw(x1: int, y1: int, x2: int, y2: int) + fill(x1: int, y1: int, x2: int, y2: int) + nextTool()

+ prevTool ()

1 + drawRect(x1: int, y1: int, x2: int, y2: int) + fillRect(x1: int, y1: int, x2: int, y2: int) + drawCircle(x: int, y: int, radius: int) + fillCircle(x: int, y: int, radius: int)

正如您所猜测的，这些是图像绘制程序的一部分。

工具集跟踪当前活动的“工具”。这是它的状态。当前设计中有三种工具:直线(默认)、矩形和圆形。然后在请求时调用Canvas中的适当方法来执行实际操作。根据活动的工具(nextTool()和prevTool())，可以告诉工具集向前或向后移动。每个工具都有能力绘图，在一个轮廓显示，也填充，在形状是坚实的着色。这对于矩形和圆是成立的。在绘制和填充线条之间没有概念上的区别，Canvas只提供了一个线条操作来满足这两种情况。

在所有情况下，当用户想要绘制或填充一些东西时，他们会从屏幕上的一点点击并拖动鼠标到另一点。(x, y)是第一次单击鼠标的位置，(x, y)是拖动鼠标的位置。1122

•对于直线，这些点表示直线的开始和结束。

•对于矩形，这些点表示矩形的两个对角。

√

•对于圆，(x, y)为圆心，半径= (x−x)+ (y−y)。111 22 1 22

你的任务如下:

(a)使用状态模式实现工具集。

提供tools set、任何接口和state子类(您自己选择)的完整代码。**(14标志)**

(b)就下列每一种情况，描述你需要如何修改你的设计。(您不需要编写任何额外的代码。(i)添加另一个运算，梯度(x1, y1, x2, y2)。假设我们已经在Canvas中添加了gradientLine()、gradientRect()和gradientCircle()。

**(ii)添加另一种表示不同形状的工具。**假设我们已经为Canvas添加了绘制和填充该形状的适当方法。

**(三是)**

**问题4出现在下一页**

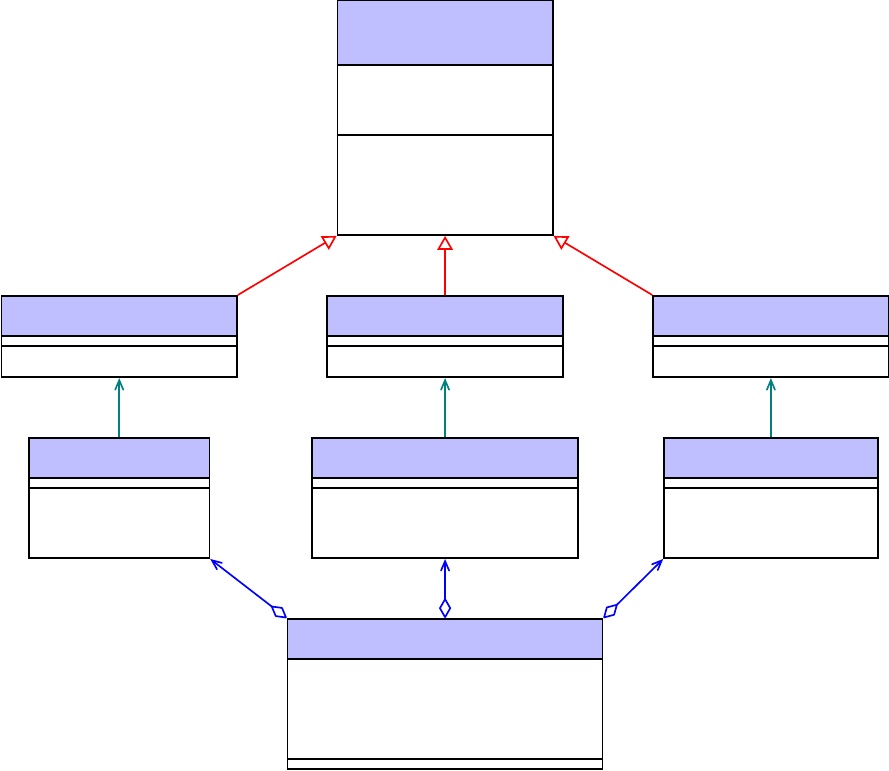
**第4页共5页**

**灰尘实得考试**

**学期2020**

**问题4(10分)**

这个问题与泛型有关。考虑下面的类结构。(这个类与上一个问题的类类似，但是两个问题之间没有联系。)



“抽象”

元素- x: int - y: int

*+ getImageData(): int[] + getX()*

砖

字符

敌人

+ getImageData(): int[

+ getImageData(): int[

+ getImageData(): int[

BrickDisplayer

CharacterDisplayer

+显示()

+ getBrick(): Brick 0..\*

+显示()

+ getCharacter():字符0..\*

GameScreen

敌人显示器()+ getEnemy():敌人0..\*

- bdisp: BrickDisplayer (0 . . \*)

- cdisp: CharacterDisplayer [0 . .\*] - edisp: EnemyDisplayer[0..\*]

砖块，角色和敌人都有一个x和y的位置，并且可以返回一个代表像素数据的整数数组。)。

Displayer类负责在屏幕上显示特定的元素类型。对于每个元素，display()方法的工作方式相同，从元素对象检索x、y和图像数据，并实际显示它们。每一个都有一个访问器，它返回相应的模型对象。(访问器很重要，这与系统设计的其余部分有关。)

识别泛型在这种情况下可以解决的问题，并相应地重构设计。解释这些更改，并使用Java或c#来显示每个受影响的类需要如何更改。(这个问题不能用Python或c++。)

您不需要编写方法或构造函数的内容。

**结束测试**

**第5页共5页**