Part 5：Grid Data Structures

被吴沛霖添加，被吴沛霖最后更新于Aug 15, 2015

Part 5: Grid Data Structures

The AbstractGrid Class

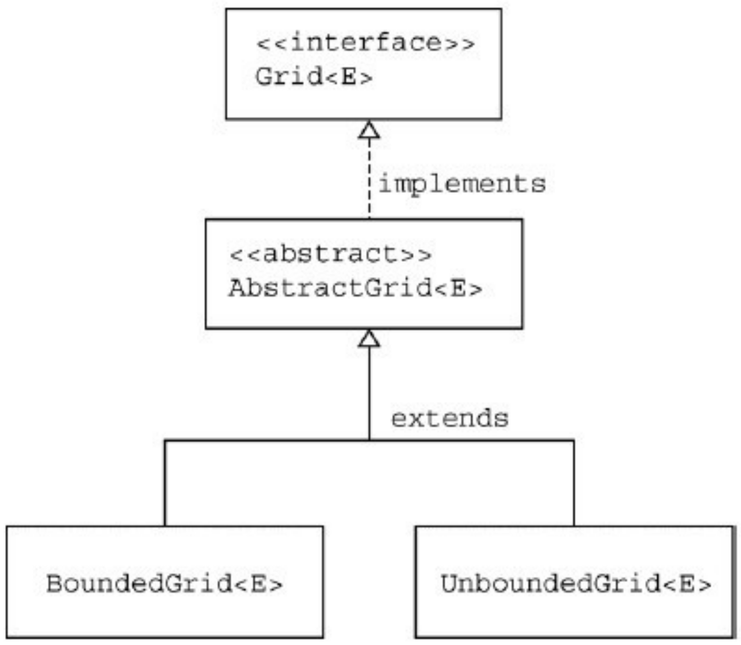
Two concrete implementations of the Grid interface are provided, one for a bounded grid that has a fixed number of rows and columns, and a second for an unbounded grid, for which any row and column values are valid. Rather than have repeated code in two classes, the AbstractGrid class defines five methods of the Grid interface that are common to both implementations.

```
public ArrayList<E> getNeighbors(Location loc)
public ArrayList<Location> getValidAdjacentLocations(Location loc)
public ArrayList<Location> getEmptyAdjacentLocations(Location loc)
public ArrayList<Location> getOccupiedAdjacentLocations(Location loc)
public String toString()
```

The code for these methods uses other methods specified in the interface. Here is the definition of the getNeighbors method. Note that the same method works for bounded and unbounded grids.

```
public ArrayList<E> getNeighbors(Location loc)
{
    ArrayList<E> neighbors = new ArrayList<E>();
    for (Location neighborLoc : getOccupiedAdjacentLocations(loc))
        neighbors.add(get(neighborLoc));
    return neighbors;
}
```

The AbstractGrid class is a superclass of the BoundedGrid and UnboundedGrid classes. Since AbstractGrid does not define all methods specified by the Grid interface, it is an abstract class. The concrete BoundedGrid and UnboundedGrid classes define the methods of the Grid interface that are not defined in AbstractGrid. This design is illustrated in the following figure.



Do You Know?

Set 10

The source code for the AbstractGrid class is in Appendix D.

1. Where is the isValid method specified? Which classes provide an implementation of this method?
2. Which AbstractGrid methods call the isValid method? Why don't the other methods need to call it?
3. Which methods of the Grid interface are called in the getNeighbors method? Which classes provide implementations of these methods?
4. Why must the get method, which returns an object of type E, be used in the getEmptyAdjacentLocations method when this method returns locations, not objects of type E?
5. What would be the effect of replacing the constant Location.HALF_RIGHT with Location.RIGHT in the two places where it occurs in the getValidAdjacentLocations method?

The BoundedGrid Class

A bounded grid has a fixed number of rows and columns. You can access only locations that are within the bounds of the grid. If you try to access an invalid location, a run-time exception will be thrown.

The BoundedGrid<E> class stores grid occupants in a two-dimensional array.

private Object[][] occupantArray;

Note that occupantArray is declared to hold references of type Object rather than the generic type E. (In the Java language, it is impossible to declare arrays of generic types.) Nevertheless, all elements of occupantArray must belong to the type E. Only the put method adds elements to the array, and it requires elements of type E.

Do You Know?

Set 11

The source code for the BoundedGrid class is in Appendix D.

1. What ensures that a grid has at least one valid location?
2. How is the number of columns in the grid determined by the getNumCols method? What assumption about the grid makes this possible?
3. What are the requirements for a Location to be valid in a BoundedGrid?

In the next four questions, let r = number of rows, c = number of columns, and n = number of occupied locations.

4. What type is returned by the getOccupiedLocations method? What is the time complexity (Big-Oh) for this method?
5. What type is returned by the get method? What parameter is needed? What is the time complexity (Big-Oh) for this method?
6. What conditions may cause an exception to be thrown by the put method? What is the time complexity (Big-Oh) for this method?

7. What type is returned by the remove method? What happens when an attempt is made to remove an item from an empty location? What is the time complexity (Big-Oh) for this method?
8. Based on the answers to questions 4, 5, 6, and 7, would you consider this an efficient implementation? Justify your answer.

The UnboundedGrid Class

In an unbounded grid, any location is valid, even when the row or column is negative or very large. Since there is no bound on the row or column of a location in this grid, the UnboundedGrid<E> class does not use a fixed size two-dimensional array. Instead, it stores occupants in a Map<Location, E>. The key type of the map is Location and the value type is E, the type of the grid occupants.

The numRows and numCols methods both return -1 to indicate that an unbounded grid does not have any specific number of rows or columns. The isValid method always returns true. The get, put, and remove methods simply invoke the corresponding Map methods. The getOccupiedLocations method returns the same locations that are contained in the key set for the map.

Do You Know?

Set 12

The source code for the UnboundedGrid class is in Appendix D.

1. Which method must the Location class implement so that an instance of HashMap can be used for the map? What would be required of the Location class if a TreeMap were used instead? Does Location satisfy these requirements?
2. Why are the checks for null included in the get, put, and remove methods? Why are no such checks included in the corresponding methods for the BoundedGrid?
3. What is the average time complexity (Big-Oh) for the three methods: get, put, and remove? What would it be if a TreeMap were used instead of a HashMap?
4. How would the behavior of this class differ, aside from time complexity, if a TreeMap were used instead of a HashMap?
5. Could a map implementation be used for a bounded grid? What advantage, if any, would the two-dimensional array implementation that

is used by the BoundedGrid class have over a map implementation?

Exercises

1. Suppose that a program requires a very large bounded grid that contains very few objects and that the program frequently calls the getOccupiedLocations method (as, for example, ActorWorld). Create a class SparseBoundedGrid that uses a "sparse array" implementation. Your solution need not be a generic class; you may simply store occupants of type Object.

The "sparse array" is an array list of linked lists. Each linked list entry holds both a grid occupant and a column index. Each entry in the array list is a linked list or is null if that row is empty.

You may choose to implement the linked list in one of two ways. You can use raw list nodes.

```
public class SparseGridNode
{
    private Object occupant;
    private int col;
    private SparseGridNode next;
    . . .
}
```

Or you can use a LinkedList<OccupantInCol> with a helper class.

```
public class OccupantInCol
{
    private Object occupant;
    private int col;
    . . .
}
```

For a grid with r rows and c columns, the sparse array has length r. Each of the linked lists has maximum length c.

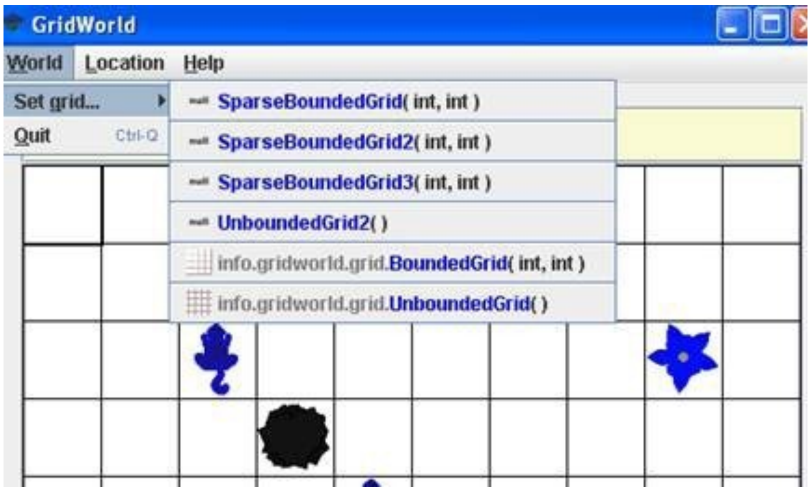
Implement the methods specified by the Grid interface using this data structure. Why is this a more time-efficient implementation than BoundedGrid?

The World has a public addGridClass method. Since the ActorWorld is a World, you can call this method in a runner. Here is the code to add a new grid to the GUI.

```
import info.gridworld.actor.Actor;
import info.gridworld.actor.ActorWorld;
import info.gridworld.grid.Location;
import info.gridworld.actor.Critter;
import info.gridworld.actor.Rock;
import info.gridworld.actor.Flower;
/**
* This class runs a world with additional grid choices.
*/
public class SparseGridRunner
{
  public static void main(String[] args)
  {
    ActorWorld world = new ActorWorld();
    world.addGridClass("SparseBoundedGrid");
    world.addGridClass("SparseBoundedGrid2");
    world.addGridClass("SparseBoundedGrid3");
    world.addGridClass("UnboundedGrid2");
    world.add(new Location(2, 2), new Critter());
    world.show();
  }
}
```

Note that you should firstly compile the SparseBoundedGrid.java to generate the SparseBoundedGrid.class.

When you execute a runner class and choose the World menu->set grid, the new grid type will be available for you to choose.


GridWorld
World
Location
Help
Set grid...
Quit Ctrl-Q
SparseBoundedGrid(int, int)
SparseBoundedGrid2(int, int)
SparseBoundedGrid3(int, int)
UnboundedGrid2( )
info.gridworld.grid.BoundedGrid(int, int)
info.gridworld.grid.UnboundedGrid( )

2. Consider using a HashMap or TreeMap to implement the SparseBoundedGrid. How could you use the UnboundedGrid class to accomplish this task? Which methods of UnboundedGrid could be used without change?

Fill in the following chart to compare the expected Big-Oh efficiencies for each implementation of the SparseBoundedGrid.

Let r = number of rows, c = number of columns, and n = number of occupied locations

Methods	SparseGridNode version	LinkedList<OccupantInCol> version	HashMap version	TreeMap version
getNeighbors				
getEmptyAdjacentLocations				
getOccupiedAdjacentLocations				
getOccupiedLocations				
get				
put				
remove				

3. Consider an implementation of an unbounded grid in which all valid locations have non-negative row and column values. The constructor allocates a 16 x 16 array. When a call is made to the put method with a row or column index that is outside the current array bounds, double both array bounds until they are large enough, construct a new square array with those bounds, and place the existing occupants into the new array.

Implement the methods specified by the Grid interface using this data structure. What is the Big-Oh efficiency of the get method? What is the efficiency of the put method when the row and column index values are within the current array bounds? What is the efficiency when the array needs to be resized?

Like Be the first to like this None

评论

张子轩 发表: Aug 20, 2015

Is there anyone want to tree tree bee with me?

庄梓嘉 发表: Aug 20, 2015

No! Uncle, we don't date!

谭潇 发表： Aug 23, 2015

活捉一只现充纸轩!

严慰 发表： Aug 22, 2015

where is Appendix D

曾坤 发表： Aug 22, 2015

code so much, heart so tired

罗思成 发表： Aug 22, 2015

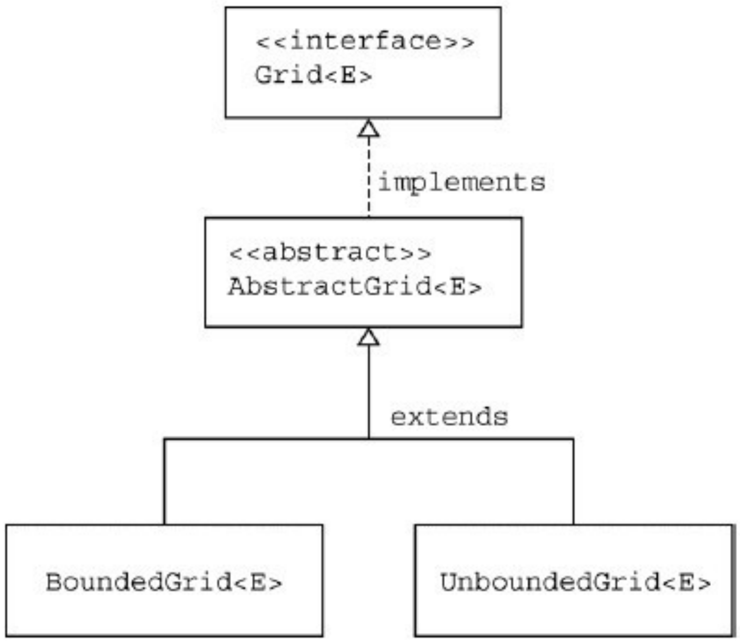
Part 5: 网格数据结构
AbstractGrid 类
提供了两个网格类接口的具体实现，一个用于有界格，其具有固定数量的行和列，还有一个是无界格，对于无界格任何行和列的值都是有效的。AbstractGrid类为两种实现定义五个通用的网格接口的方法，而不是在两个类中写重复的代码

```
--------------------------------------------------
public ArrayList<E> getNeighbors(Location loc)
public ArrayList<Location> getValidAdjacentLocations(Location loc)
public ArrayList<Location> getEmptyAdjacentLocations(Location loc)
public ArrayList<Location> getOccupiedAdjacentLocations(Location loc)
public String toString()
--------------------------------------------------
```

对于这些方法的代码使用接口中指定的其他方法。这里是getNeighbors方法的定义。需要注意的是同样的方法适用于有界和无界网格。

```
--------------------------------------------------
public ArrayList<E> getNeighbors(Location loc)
{
ArrayList<E> neighbors = new ArrayList<E>();
for (Location neighborLoc : getOccupiedAdjacentLocations(loc))
neighbors.add(get(neighborLoc));
return neighbors;
}
--------------------------------------------------
```

该AbstractGrid类是BoundedGrid和UnboundedGrid类的父类。因为AbstractGrid没有定义由网格接口中指定的所有方法，它是一个抽象类。具体的方法由BoundedGrid和UnboundedGrid类定义（也就是未在AbstractGrid定义的网格接口的方法）。设计说明如下图。



你知道吗?

Set 10

AbstractGrid 类源代码在附录D
1.isValid方法在那里被声明？哪个类提供此方法的具体实现?
2.AbstractGrid哪些方法调用了isValid方法？为什么其他的方法不需要调用isValid?
3.网格接口中的哪些方法在getNeighbors中被调用？哪个类提供了这些方法的具体实现?
4.为什么get方法(返回类型为E)要在getEmptyAdjacentLocations方法中被使用（此方法返回location，而不是E）？
5.将getValidAdjacentLocations的两处Location.HALF_RIGHT用常量Location.RIGHT替换会有什么效果?

BoundedGrid类
有界格拥有固定的行数和列数。你只能访问边界内的位置。如果您尝试访问无效的位置，运行时会抛出异常。

BoundedGrid<E> 类用二维数组存储网格的占有者
例如，下列数组：
private Object[][] occupantArray;

需要注意的是occupantArray是用来存储 Object 的引用的，而不是泛型E 的引用。（在Java语言中，声明泛型数组是不可能的）

然而，occupantArray的所有元素的类型都是E.只用PUT方法为它添加元素，而它只添加类型 E 的元素

你知道吗?

Set 11

oundedGrid类的源代码在附录D

1.什么确保了网格有至少一个有效的位置?
2.getNumCols方法是如何确定列数的？在哪些有关grid假定的前提下getNumCols方法能确定列数
3.BoundedGrid中合法位置有什么要求?

在接下来的四个问题中，假设 r = 行数，c =数，n = 占用的位置数

4.getOccupiedLocations方法返回什么类型？他的时间复杂度（大O）是多少?
5.GET方法返回什么类型？需要什么参数？时间复杂度（大O）是多少?
6.什么情况可能会导致put方法抛出异常？时间复杂度（大O）是多少?
7.remove方法返回什么类型？当试图从一个空location删除项目时会发生什么？时间复杂度（大O）是多少?
8.根据问题4，5，6，7的答案，你认为这是一个高效的实现吗？证明你的答案。

UnboundedGrid类
在无界格中，任何位置都是有效的，即使是负数或是非常大的数。由于网格中的位置没有边界，UnboundedGrid <E>类不使用固定大小的二维数组。相反，它在 Map<Location，E> 中存储网格的占有者。Map<Location，E>的键是 Location，值为 E，即网格的占有者的类型

UnboundedGrid类中，numRows 和 numCols 方法均返回-1（说明没有固定行数和列数）。isValid方法总是返回true。get，put和remove方法简单地调用相应的 Map<Location，E> 的方法。getOccupiedLocations方法返回一个包含在Map<Location，E>上键集合的相同位置。

你知道吗?

Set 12

UnboundedGrid类的源代码在附录D

1.Location类必须实现哪些方法才能使得一个HashMap的实例能用来当做地图？如果是TreeMap要实现哪些方法呢？ 现有的Location 是否满足这些要求?
2.为什么get, put, 和 remove 的方法都包含检查值为 null呢？为什么BoundedGrid类中这三个函数没有这样的检查?
3.get，put 和remove 的平均时间复杂度（大O）分别是多少？如果用TreeMap中来代替HashMap的话，又是多少?
4.如果用TreeMap来代替HashMap的话，除了时间复杂度之外，UnboundedGrid类有哪些行为会不同?
5.UnboundedGrid类的 map 实现能用在有界网格吗？BoundedGrid类二维数组的做法比起UnboundedGrid类的 map 有什么好处

练习
1.假设一个程序需要一个非常大的有界格却包含非常少的对象，并且该程序经常调用getOccupiedLocations方法（例如，ActorWorld）。创建一个SparseBoundedGrid类（使用“疏阵”实现）。您的解决方案不必是通用类;你可以将网格的占有者用Object类型存储。

而“疏阵”是元素为链表的数组。每个链表元素包含网格的占有者和列索引。数组中的每个元素是一个链表或者是 null（该行为空）

你可以选择下列两种方法中的一种来实现，定义一种 list node

```
--------------------------------------------------
public class SparseGridNode
{
private Object occupant;
private int col;
private SparseGridNode next;
. . .
}
--------------------------------------------------
```

或者实现一个辅助类来使用LinkedList<OccupantInCol>

```
--------------------------------------------------
public class OccupantInCol
{
private Object occupant;
private int col;
. . .
}
--------------------------------------------------
```

对于具有r行和c列的网格，sparse array 的 length 为 r。 每个链表的最大长度为 c

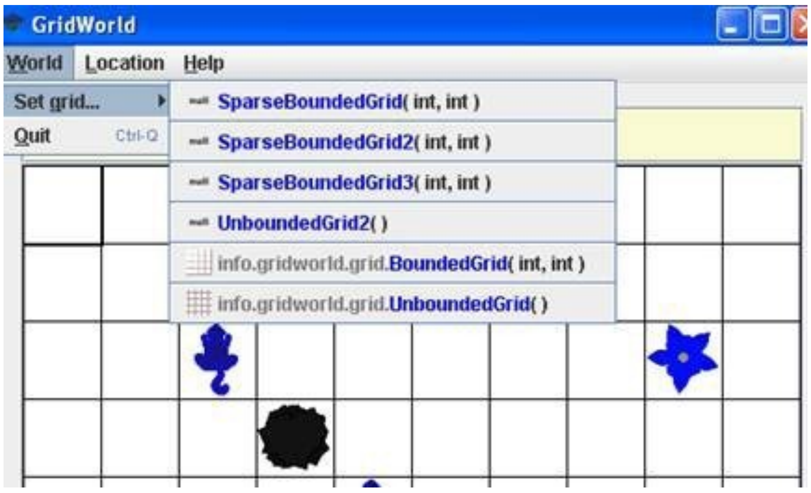
使用这样的数据结构通过网格接口来实现特定的方法，为什么会比BoundedGrid更省时和有效?

World 有一个公共的addGridClass方法。由于ActorWorld是一个 World，你可以 在 Runner 中调用此方法在一个亚军。参考代码如下

```
--------------------------------------------------
import info.gridworld.actor.Actor;
import info.gridworld.actor.ActorWorld;
import info.gridworld.grid.Location;
import info.gridworld.actor.Critter;
import info.gridworld.actor.Rock;
import info.gridworld.actor.Flower;
/**
* This class runs a world with additional grid choices.
*/
public class SparseGridRunner
{
public static void main(String[] args)
{
ActorWorld world = new ActorWorld();
world.addGridClass("SparseBoundedGrid");
world.addGridClass("SparseBoundedGrid2");
world.addGridClass("SparseBoundedGrid3");
world.addGridClass("UnboundedGrid2");
world.add(new Location(2, 2), new Critter());
world.show();
}
}
--------------------------------------------------
```

请注意，您应该先编译SparseBoundedGrid.java来产生SparseBoundedGrid.class。

当你执行一个 runner 类，选择 menu->set 设置网格就可以选择新的网格类型啦-


GridWorld
World
Location
Help
Set grid...
Quit Ctrl-Q
SparseBoundedGrid(int, int)
SparseBoundedGrid2(int, int)
SparseBoundedGrid3(int, int)
UnboundedGrid2( )
info.gridworld.grid.BoundedGrid(int, int)
info.gridworld.grid.UnboundedGrid( )

2.考虑使用HashMap或TreeMap来实现SparseBoundedGrid类。怎样能使用UnboundedGrid类来完成这项任务? UnboundedGrid类中哪些方法可以不用改动?

填写下表来比较 SparseBoundedGrid 类的每种实现的时间复杂度（大O）

r = number of rows, c = number of columns, and n = number of occupied locations

Methods	SparseGridNode version	LinkedList<OccupantInCol> version	HashMap version	TreeMap version
getNeighbors				
getEmptyAdjacentLocations				
getOccupiedAdjacentLocations				
getOccupiedLocations				
get				
put				
remove				

3.来看看无界格，他的构造函数分配了一个16x16的数组。当put方法被调用来创建当前数组边界之外的元素，double当前的列数和行数，直到它们满足要求，构造一个新的方阵数组，并把现有的网格占有者迁移到新数组。

使用这样的数据结构来通过网格接口实现特定的方法。 get 方法的时间复杂度（大O）是多少？put方法呢？（行和列的索引值在当前的阵列范围之内）？ 当数组需要调整大小时的时间复杂度呢？

孙乾晟 发表： Aug 23, 2015

好人一生平安

曾坤 发表： Aug 23, 2015

思成哥，为什么最后那个非负数没有翻译出来啊！

邱永臣 发表： Aug 23, 2015

全是大神队友，基本不用打码😊