题目描述

本次题目要求你构建一个快递管理系统，实现寄件、入库、取件、查询等功能。为了实现这些功能，你一共需要实现4个类：包裹类(Package)类，普通包裹类(RegularPackage)，贵重包裹类(ValuablePackage)，快递点类(CourierPoint)。其中RegularPackage类和ValuablePackage类是Package类的派生类，表示两种类型的快递包裹(两种包裹的差别主要体现在运费计算方式不同)；CourierPoint类表示不同的快递站点。

- Package类
  - 你需要在自己定义的Package类中至少实现以下接口

```
# 初始化函数，初始化Package对象
def __init__ (self, package_id: str)
"""
说明
1) package_id表示包裹id
2) 不同的包裹的id是不同的
"""

# 实例方法，返回该包裹的id
def GetPackageID(self) -> str

# 实例方法，计算包裹的运送费用并返回
# 在基类Package中不需要具体实现计算，但是需要抛出自定义异常CustomError，并设定异常描述为"调用异常"
def ComputeDeliverPrice(self, dist: int)
"""
说明
dist为运送距离
"""
```

  - 接口调用实例

```
package0 = Package("1000")

try:
  package0.ComputeDeliverPrice(100)
except CustomError as e:
  print(e) # 输出 "调用异常"
```

- RegularPackage类
  - RegularPackage类是Package类的派生类，你需要在自己定义的RegularPackage类中具有以下接口

```python
# 初始化函数，初始化RegularPackage对象
def __init__(self, package_id: str)

# 实例方法(继承自基类)，返回该包裹的id
def GetPackageID(self) -> str

# 实例方法，计算该包裹的运送费用并返回
def ComputeDeliverPrice(self, dist: int) -> int
"""
说明
运送费用 = 运送距离 * 1
"""
```

- 接口调用示例

```python
# 创建对象
package1 = RegularPackage("1001")
# 查询包裹id
package1.GetPackageID()
# 计算运费
package1.ComputeDeliverPrice(300)
```

- ValuablePackage类
  - ValuablePackage类是Package类的派生类，你需要在自己定义的ValuablePackage类具有以下接口

```python
# 初始化函数，初始化ValuablePackage对象
def __init__(self, package_id: str)

# 实例方法函数(继承自基类)，返回该包裹的id
def GetPackageID(self) -> str

# 实例方法，计算该包裹的运送费用并返回
def ComputeDeliverPrice(self, dist: int) -> int
"""
说明
运送费用 = 运送距离 * 5
"""
```

  - 接口调用示例

```
# 创建对象
package2 = ValuablePackage("1002")
# 查询包裹id
package2.GetPackageID()
# 计算运费
package2.ComputeDeliverPrice(300)
```

- CourierPoint类
  - 你需要在自己定义的CourierPoint类中至少实现以下接口

```
# 初始化函数，初始化CourierPoint对象
def __init__(self, x: int, y: int)
"""
说明
1) (x, y)为该快递点的坐标位置，x和y均大于等于0
2) 不用考虑在同一坐标位置有多个快递点的情况
3) 快递点的仓库最多可存储的10个包裹
"""

# 静态方法，计算曼哈顿距离并返回
@staticmethod
def ComputeManhhanDistance(start_x: int, start_y: int, target_x: int, target_y: int) -> int

# 实例方法，寄包裹并返回运送费用
def Send(self, package: Package, dest_x: int, dest_y: int) -> int
"""
说明
1) (dest_x, dest_y)为寄送目的地的坐标位置
2) 假设包裹需要从起始快递点(start_x, start_y)寄往目的地(dest_x, dest_y)，则首先需要从已有的快递点集合中寻找距离目的地最近的快递点(target_x, target_y)，计算出起始快递点(start_x, start_y)到目标快递点(target_x, target_y)之间的曼哈顿距离，再将曼哈顿距离作为参数传入包裹的计算运费函数，得出运送费用
3) 以下情况视为寄送失败，返回"-1"：1. 若距离目的地最近的快递点就是起始快递点；2. 目标快递点容量已满；
4) 如果没有发生错误，则将该包裹投递到目标快递点。
5) 不需要考虑以下特殊情况：距离目的地最近的快递点除了起始快递点以外，还有两个及以上的快递点距离目的地最近且距离相同
"""

# 实例方法，包裹入库并返回执行状态
def Receive(self, package: Package) -> bool
"""
说明
当仓库容量已满时，则入库失败，仓库中已有包裹保持不变，返回False；当仓库容量仍有剩余时，则入库成功，返回True
"""
```

```
31
32  # 实例方法，查询仓库中的包裹并返回包裹对象
33  def Find(self, package_id: str) -> Package
34  """
35  说明
36  若未找到包裹id为package_id的包裹，则返回None；若找到包裹id为package_id的包裹，则返回包裹
37  """
38
39  # 实例方法，取出包裹并返回执行状态
40  def Pick(self, package_id: str) -> bool
41  """
42  说明
43  若仓库中没有包裹id为package_id的包裹，则取件失败，返回False；若仓库中有包裹id为package_id的包裹，则取件成功，将此包裹从仓库中移出，返回True
44  """
```

- 接口调用示例

```
1   # 创建对象
2   courier_point1 = CourierPoint(0, 0);
3   courier_point2 = CourierPoint(10, 0);
4   courier_point3 = CourierPoint(20, 0);
5   package3 = ValuablePackage("1003");
6   package4 = RegularPackage("1004");
7
8   # 计算曼哈顿距离
9   CourierPoint.ComputeManhhanDistance(0, 0, 10, 0)
10
11  # 寄包裹：目的地坐标为(12, 0)，起始快递点坐标为(0, 0)，在已有的三个快递点中距离(12, 0)的曼哈顿距离最小的是目标快递点(10, 0)，所以运送距离为|0-10|+|0-0|=10，再根据距离10计算运送费用并返回
12  courier_point1.Send(package3, 12, 0)
13
14  # 取包裹：由于package3被寄到了courier_point2中，这时可以从courier_point2中取出包裹
15  courier_point2.Pick("1003")
16
17  # 包裹入库
18  courier_point1.Receive(package4)
19
20  # 查询包裹
21  courier_point1.Find("1004")
22
23  # 取出包裹
24  courier_point1.Pick("1004")
```

注意⚠️

- 提交格式

```
# 在此处填写代码
# your code here

# 测试用代码，请不要改动
while True:
    exec(input())
```

扩展阅读📖

- 曼哈顿距离：https://zh.wikipedia.org/wiki/曼哈頓距離