OOP Basics Exam: It's in the Blood

1. Overview

Have you ever thought about why are you getting sick? When it comes to harmful agents, your body turns into a real battlefield. The healthier your body is, the faster you get better. Now you are going to simulate the battle against the some harmful microbes, trying to get you sick. You are armed with your blood cells, and some boosters, that make you stronger and healthier.

2. Structure

On top of the structure of our simulation stands the **organism** – you may say this is your body. This organism has formations, called **clusters** and the clusters consist of different **cells**.

Cell

Cells have 4 properties you must implement:

- **`id –`** a **String**, **nonunique** property**;**
- **`health`** – a **positive integer**;
- **`positionRow`** – a **positive integer**;
- **`positionCol`** – a **positive integer**;

All these properties are set trough the constructor. You will find out what is this position about very soon.

There are **2 major types** of cells – **Blood cells** and **Microbes**

BloodCell

There are **2 types** of blood cells – **White** and **Red** blood cells.

WhiteBloodCell

All **white blood cells** have **1 additional property** to implement:

- **`size`** – a **positive integer** that describes the **size** of the cell.

The size should be set trough the constructor.

RedBloodCell

All **red blood cells** have **1 additional property** you must implement:

- **`velocity`** – a **positive integer** that describes the **speed** of the cell.

The velocity should be set trough the constructor.

Microbe

All microbes have 1 additional property:

- **`virulence`** – a **positive integer** that describes the ability of a microbe to infect other cells.

The virulence should be set trough the constructor.

There are 3 types of microbes – **Bacteria**, **Virus** and **Fungi**

Cluster

Clusters are some kind of blocks of cells. They have 4 properties you must implement (all set through the constructor):

- **`id`** – a **string** that describes the **name** of the cluster, unique property;
- **`rows`** – a **positive integer** that describes how many **rows** the cluster has;
- **`cols`** – a **positive integer** that describes how many **columns** the cluster has;
- **`cells`** – a **collection of objects** of type **`Cell`**;

Organism

The organism has 1 property set trough the constructor:

- **`name`** – a string that describes the **name** of the **`Organism`**;

And one additional property:

- **`clusters`** – a **collection of objects** of type **`Cluster`**;

Constructors

Implement all **class constructors**, with the **parameters** in the **EXACT given order** and the **EXACT given types (use primitive types as parameters)**. Properties of derived classes should come after their predecessors'. All constructors described above should be public.

String Representation

Implement **`toString()`** methods for the Organism, Cluster and Cell classes (this is necessary for testing purposes). You can see the requirements in the **Output Section** below.

3. Business Logic

Testing

You can gain points from this task just after you implement correctly **checkCondition()** and at least one of the other methods correctly.

The Controller Class

The business logic of the program should be concentrated around several commands. Implement a class called **`HealthManager`**, which will hold the **main functionality**, represented by **EXACTLY** these **methods** (provide methods with correct return types, names and parameters):

- **`String checkCondition(String organismName)`**
- **`String createOrganism(String name)`**
- **`String addCluster(String organismName, String id, int rows, int cols)`**
- **`String addCell(String organismName, String clusterId, String cellType, String cellId, int health, int positionRow, int positionCol, int additionalProperty)`**
- **`String activateCluster(String organismName) All given methods should be public.`**

Commands

The commands in the **HealthManager class** should represent the **functionality** to the input commands of the user. Here are the **input commands** you need to accept from the **user input**.

- **checkCondition {OrganismName}**
  - **RETURNS** detailed information about the condition of the **organism** with the given **name**
- **createOrganism {Name}**
  - **CREATES** an organism with the given **name**
  - **RETURNS** message "**`Created organism <name>`**"
  - If an organism with the **same name already exists**, **returns** message "**`Organism <name> already exists`**"
- **addCluster {organismName} {id} {int rows} {int cols}**
  - **CREATES** a cluster with the given **id**, **rows** and **cols**
  - **ADDS** the cluster to the cluster collection of the **organism** with the given **name**
  - If the **organism** already **has a cluster** with the **same Id**, **nothing** happens
  - **RETURNS** message "**`Organism <organism name>: Created cluster <cluster id>`**";
- **addCell {OrganismName} {ClusterId} {CellTYpe} {CellId} {health} {positionRow} {positionCol} {additionalProperty}**
  - **CREATES** a **cell** of the given **type** with the given **id**, **health**, **positionRow**, **positionCol**, and the given **additional property (size, velocity or virulense).**
  - **FINDS** the **organism** with given **name**, find the **cluster** with given **id** in the **cluster collection** of that organism and **ADDS** the cell to the **cells collection** of that cluster
  - **RETURNS** message **"`Organism <organism name>: Created cell <cell id> in cluster <cluster id>`"**
- **activateCluster {Name}**
  - **FINDS** the **organism** with the given **name**
  - **ACTIVATES** the next cluster in order
  - **RETURNS** message "**`Organism <organism name>: Activated cluster <cluster id>. Cells left: <cells count>`**"

Functionality

Entity Creation

Health manager is responsible for organism, cluster and cell creation so it should produce correct output for each new entity created.

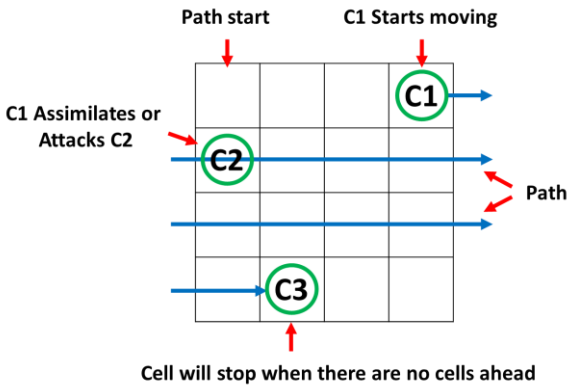
In case of invalid input such as **nonexistent organism** name, **nonexistent cluster** name, cell position **outside of a cluster** or creation attempt for a **duplicate cluster id** inside of the same organism, the **command should be ignored**.

In case of creation of an organism that **already exists**, print: "**Organism <name> already exists**".

Cluster Activation

Clusters are always activated in order of creation. After a cluster is activated, it **should wait for all the other clusters** until it can be activated again.

Cluster activation chooses the cell closest to the start point of the cluster and moves it first to its right along a strict path. The **start point** is always at coordinates **[0,0].** The cell that moves is the **first along the path**. The paths direction is always right and when the end of a row is reached, the path continues at the start of the **next row.** The cell stops when there are **no more cells ahead on the path** (see the picture).



Moving through the cluster the cell meets other cells. Blood cells and Microbes have different behaviour when this happens.

When **Blood cell** meets **another Blood cell or Microbe**, it **assimilates** it and **takes** all of its **health**. The **other cell is assimilated and disappears.** The moving cell continues its way.

When a **Microbe** meets another **Microbe** or a **Blood cell**, it **attacks**. When the cell **attacks**, the target **takes damage**, equal to the **energy of the attacking cell**. Each cell has energy and it depends on the **other stats** of the cell.

- **RedBloodCell: energy = health + velocity.**
- **WhiteBloodCell: energy = (health + size) * 2**.
- **Bacteria: energy = (health + virulence) / 3.**
- **Fungi: energy = (health + virulence) / 4.**
- **Virus: energy = health + virulence.**

After the attack, If the **target is still alive**, it **strikes back**. This goes on until one of the cells has 0 health. Then it should be removed from the cluster. The cell that survives continues along the path.

The moving cell stops when there are no more cells ahead. For example if a cluster with only one cell is activated, it will not change its state.

Input/Output

Input

- The input will come in the form of commands, in the format specified above.
- The input sequence ends when you receive the command "**BEER IS COMING**".

Output

Each command prints a text result or nothing if there are invalid parameters:

- createOrganism:
  - " **Created organism {organism name}**" or
  - "**Organism already exists**"
- addCluster: "**Organism {organism name}: Created cluster {cluster name}**"
- addCell: "**Organism {organism name}: Created cell {cell name} in cluster {cluster name}**"
- activateCluster: " **Organism {organism name}: Activated cluster {cluster name}. Cells left: {cells left}**"
- The "**checkCondition**" **RETURNS** a **String representation** of the **ORGANISM** with the **GIVEN NAME**:
  - **"Organism - {name}**

- **--Clusters: {clusters count}**
- **--Cells: {cells count}**
- **----Cluster {clusterId}**
- **------Cell {id} [{positionRow},{positionCol}]**

If the cell is WhiteBloodCell, you should print:

- **--------Health {health} | Size {size} | Energy {energy}**

If the cell is RedBloodCell, you should print:

- **--------Health {health} | Velocity {velocity} | Energy {energy}**

If the cell is a Microbe, you should print:

- **--------Health {health} | Virulence {virulence} | Energy {energy}"**

Cells should be **ordered by** positionRow in ascending order, then by positionCol in ascending order

Constrains

- **All integers** in the input will be in **range [0, 100000]**.
- **All strings** in the input may consist of **any ASCII character**, except **SPACE**
  - So that the input is easily processed.
- There will be **NO** cells with the **same positionRow AND** the **same positionCol** properties.
- Note that throughout the program, you are working **ONLY** with **INTEGERS**.
  - Each **mathematical** or **logical action** performed on **numeric data**, should be performed between **INTEGERS**.

Examples

Input	Output
createOrganism Troli createOrganism Troli addCluster Trr X05 2 3 addCell Trr X05 WhiteBloodCell WBC 5 0 0 5 checkCondition Troli addCluster Troli X05 2 3 addCell Troli X05 WhiteBloodCell WBC 5 0 0 5 checkCondition Troli BEER IS COMING	Created organism Troli Organism Troli already exists Organism - Troli --Clusters: 0 --Cells: 0 Organism Troli: Created cluster X05 Organism Troli: Created cell WBC in cluster X05 Organism - Troli --Clusters: 1 --Cells: 1 ----Cluster X05 ------Cell WBC [0,0] --------Health: 5 | Size: 5 | Energy: 20
createOrganism Troli addCluster Troli X05 2 3 addCell Troli X05 WhiteBloodCell WBC 50 0 0 20 addCell Troli X05 RedBloodCell RBC 30 1 2 18 addCell Troli X05 Virus V 100 1 1 10 activateCluster Troli checkCondition Troli BEER IS COMING	Created organism Troli Organism Troli: Created cluster X05 Organism Troli: Created cell WBC in cluster X05 Organism Troli: Created cell RBC in cluster X05 Organism Troli: Created cell V in cluster X05 Organism Troli: Activated cluster X05. Cells left: 1 Organism - Troli --Clusters: 1 --Cells: 1 ----Cluster X05 ------Cell WBC [1,2]

	--------Health: 180 | Size: 20 | Energy: 400
createOrganism Ivan createOrganism Gosho addCluster Ivan H8 2 3 addCell Ivan H8 Bacteria B 150 0 0 10 addCell Ivan H8 RedBloodCell RBC 60 1 2 10 addCell Ivan H8 WhiteBloodCell WBC 10 1 1 10 addCluster Ivan H9 2 3 addCell Ivan H9 Virus V1 10 0 0 10 addCell Ivan H9 Fungi F1 30 0 1 20 addCell Ivan H9 WhiteBloodCell WBC 100 1 2 30 addCell Ivan H9 RedBloodCell RBC 100 1 1 20 activateCluster Ivan checkCondition Ivan checkCondition Gosho BEER IS COMING	Created organism Ivan Created organism Gosho Organism Ivan: Created cluster H8 Organism Ivan: Created cell B in cluster H8 Organism Ivan: Created cell RBC in cluster H8 Organism Ivan: Created cell WBC in cluster H8 Organism Ivan: Created cluster H9 Organism Ivan: Created cell V1 in cluster H9 Organism Ivan: Created cell F1 in cluster H9 Organism Ivan: Created cell WBC in cluster H9 Organism Ivan: Created cell RBC in cluster H9 Organism Ivan: Activated cluster H8. Cells left: 1 Organism - Ivan --Clusters: 2 --Cells: 5 ----Cluster H9 ------Cell V1 [0,0] --------Health: 10 | Virulence: 10 | Energy: 20 ------Cell F1 [0,1] --------Health: 30 | Virulence: 20 | Energy: 12 ------Cell RBC [1,1] --------Health: 100 | Velocity: 20 | Energy: 120 ------Cell WBC [1,2] --------Health: 100 | Size: 30 | Energy: 260 ----Cluster H8 ------Cell B [1,2] --------Health: 133 | Virulence: 10 | Energy: 47 Organism - Gosho --Clusters: 0 --Cells: 0