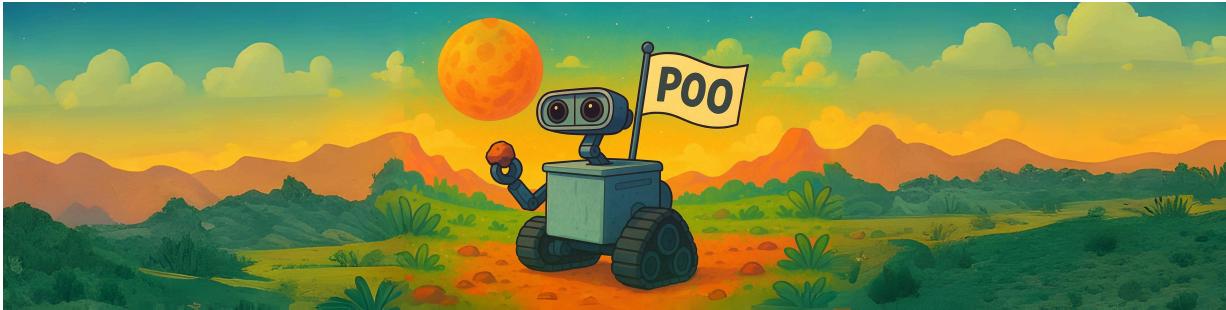


## TerraBot - let's explore and save the planet !

---



- **Responsabili & Autori:** Alina Tudorache [mailto:alina.tudorache872@gmail.com], Dragos Dragan [mailto:dragos.o.dragan@gmail.com], Andrei Marcu [mailto:marcu.andrei.nectarie@gmail.com], Alexia Oprisan [mailto:alexiops2014@gmail.com], Francesco Martinuț [mailto:francescomartinut18@gmail.com], Cleopatra Popescu [mailto:cleopatra.popescu02@gmail.com]
- **Consultanți & Revizori:** Sorina-Anamaria Buf [mailto:sorinabuf@gmail.com], Ștefan Cocioran [mailto:stefancocioran@gmail.com], Florian-Luis Micu [mailto:miculuis1@gmail.com]
- **Data publicării BETA testing:** 21.10.2025
- **Deadline HARD BETA testing:** 15.11.2025, ora 10:00
- **Data publicării oficiale:** 10.11.2025
- **Deadline HARD oficial:** 30.11.2025
- **Ultima modificare (cerință, schelet sau teste):**
  - enunț
    - update formulă mean la Deplasare robot - 26 oct. 2025
    - update nou mesaj eroare pentru Comenzi robot - 26 oct. 2025
    - update valoare scor la Deplasare robot - 26 oct. 2025
    - update specificații Git - 26 oct. 2025
    - update formulă normalizare scor calitate aer și sol - 27 oct. 2025
    - adăugat explicații scanObject și mișcare animal - 28 oct. 2025
    - adăugat explicații exemplu simulare, toxicitate aer, improveEnvironment, changeWeatherConditions, rechargeBattery, entitate Water, interacțiuni între entități - 4 nov. 2025
    - adăugat explicații Algoritm hrănire animal - 10 nov. 2025
    - adăugat explicații Algoritm hrănire animal - 13 nov. 2025
    - adăugat explicații desertStorm pentru printEnvConditions - 18 nov. 2025
    - update specificare sol improveEnvironment - 27 nov. 2025
  - schelet
    - update refs t3, t4 - 26 oct. 2025
    - update TestRunner + adăugat t11-16 - 26 oct. 2025
    - update t8 + ref - 27 oct. 2025
    - update refs + clasa WaterInput + t17,18 - 4 nov. 2025
    - adăugat teste complexe t19-21 - 4 nov. 2025
    - patch schelet - 8 nov. 2025
    - update refs t15, 19-21 - 9 nov. 2025
    - update t20,21 + ref t20 - 10 nov. 2025
    - update t19 + refs t8,19 - 12 nov. 2025
    - update refs t8,19,21 - 13 nov. 2025
  - teste
    - adăugat t11-16 - 26 oct. 2025
    - update t8 - 27 oct. 2025
    - update t5,7-11, 16 + adăugat t17,18 - 4 nov. 2025
    - adăugat teste complexe t19-21 - 4 nov. 2025
    - update refs t15, 19-21 (stergere planta) - 9 nov. 2025
    - update t20,21 + ref t20 - 10 nov. 2025
    - update t19 + refs t8,19 - 12 nov. 2025
    - update refs t8,19,21 - 13 nov. 2025
- **Schelet și teste:** <https://github.com/oop-pub/schelet-tema1-2025> [<https://github.com/oop-pub/schelet-tema1-2025>]

Soluția voastră trebuie implementată folosind **principiile programării orientate pe obiecte (OOP)**, nu în stil imperativ, ca în limbajul C.

Scopul exercițiului este să **demonstrezi înțelegerea conceptelor OOP**, nu doar să obținești un rezultat funcțional.

Dacă aveți 100p pe checker, însă soluția voastră nu este OOP, atunci veți obține o **depunctare majoră** conform baremului public!

Pentru detalii consultați acest ghid care conține **baremul de corectare** și alte **guidelines**.

- Regulamentul pentru perioada de dezvoltare BETA îl găsiți [aici](#).
- Informațiile de tip **[Nice to know]** din laborator vă pot ajuta la depanare sau la rezolvarea temei mai eficient.
- Tema a fost concepută pentru a fi rezolvată folosind **laboratoarele 1-6**, dar puteți folosi concepte și din laboratoarele următoare.
- Pentru perioada BETA, puteți folosi **laboratoarele din anii trecuți** pentru a găsi diverse informații, dar vă recomandăm să aprofundați **două** următoarele concepte pentru această temă:
  - Sintaxa Java.
  - Clase și obiecte.
  - Moștenire și polimorfism.
  - Interfețe și clase abstracte.
  - Colecții.
  - Clase interne.
  - Exceptii.
  - Static.
  - Immutabilitate.

## Obiective

---

- Familiarizarea cu Java și conceptele de bază ale POO.
- Fundamentarea practică a constructorilor și a agregării/moștenirii.
- Dezvoltarea unor abilități de bază de organizare și design orientat obiect.
- Scrierea unui cod cât mai generic, ce va permite ulterior adăugarea de noi funcționalități.
- Familiarizarea conceptelor de polimorfism, abstractizare și încapsulare.
- Utilizarea colecțiilor din Java.
- Folosirea claselor speciale și a bibliotecii standard Java.
- Respectarea unui stil de codare și de comentare.
- Dezvoltarea aptitudinilor de depanare a problemelor în cod.

## Introducere

---

TerraBot, robotul nostru explorator creat de NASA, are o misiune deosebită: să descopere o nouă planetă și să o pregătească pentru viitoarea colonizare. El trebuie să exploreze cât mai mult din suprafața planetei, să adune informații esențiale despre mediul înconjurător și să contribuie la îmbunătățirea condițiilor de viață. Totuși, fiți atenți, deoarece energia lui TerraBot este limitată, așa că fiecare acțiune trebuie planificată cu grijă pentru a garanta succesul misiunii!

## Misiunea lui TerraBot

---

Odată lansat în misiune, TerraBot intră într-un mediu complet necunoscut, unde fiecare regiune a planetei ascunde provocări și condiții diferite. Robotul nostru este programat să efectueze simulări experimentale, fiecare având scopul de a observa cum evoluează ecosistemul, de a colecta date despre plante, animale, apă și sol, și de a aplica diferite metode pentru a menține mediul sănătos și sustenabil.

Pe parcursul fiecărei simulări, TerraBot experimentează, analizează datele obținute și ia decizii autonome pentru a-și îndeplini obiectivele. O simulare reprezintă o rundă completă de explorare: mediu se modifică în timp real, resursele sunt limitate, iar robotul trebuie să acționeze eficient și adaptabil. Astfel, în cadrul testelor pot exista una sau mai multe simulări, permățându-vă să evaluați performanța TerraBot în scenarii diferite.

Fiecare test are următoarea structură:

1. Se generează preambulul – se creează harta planetei, cu toate entitățile poziționate pe celule (vezi secțiunea *Desfășurarea simulării*).
2. Se inițiază simularea.
  - comanda **startSimulation** – marchează începutul simulării.
3. Se desfășoară simularea propriu-zisă – TerraBot primește comenzi, reacționează la mediu și se actualizează starea planetei.
4. Se finalizează simularea.
  - comanda **endSimulation** → marchează finalul simulării.
5. Se repetă pașii de mai sus pentru fiecare simulare indicată în cadrul testului.

## Desfășurarea unei simulări

---

Fiecare simulare trebuie să înceapă cu comanda **startSimulation** și să se încheie cu **endSimulation**.

### 1. Evoluția automată a mediului

Se produce independent de acțiunile TerraBot și reflectă procese naturale:

- plantele cresc sau se ofilesc
- animalele se deplasează și își schimbă starea (consumă plante sau apă)
- calitatea aerului, solului și apei se modifică în timp (în această ordine)

### 2. Acțiunile TerraBot

Se execută pe baza comenzielor trimise:

- se deplasează pe hartă
- scană mediul și învață informații (facts)
- colectează plante, animale sau apă în inventar
- aplică metode de îmbunătățire a mediului
- explorează noi zone în funcție de calitatea mediului

Ce este o iterație?

O simulare este împărțită în iterări (unități de timp).

La fiecare iterare se execută **ambele** procese în **această ordine**: evoluția mediului și acțiunile robotului.

**1 iterare = 1 timestamp**

O simulare este considerată validă doar dacă toate iterările sunt corect procesate.

Doar comenzi aflate între aceste două marcaje sunt considerate **valide** și vor fi **executate**. Prin urmare, o simulare trebuie să aibă atât un **început**, cât și un **sârșit**.

Condițiile inițiale ale simulării

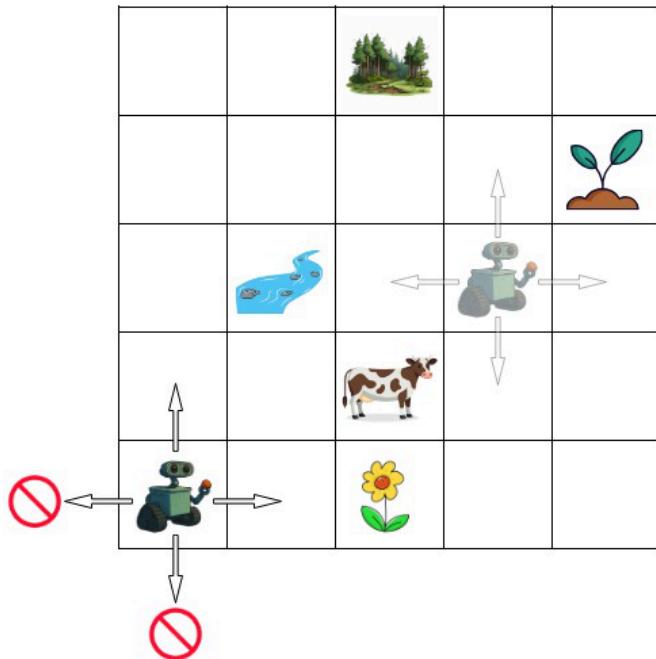
La începutul fiecărei simulări:

- TerraBot pornește de la zero:
  - bateria este complet încărcată
  - inventarul este gol
- Mediul este resetat:
  - harta este regenerată cu parametri noi
  - fiecare celulă primește un tip de sol
  - sunt repozitionate plantele și animalele
  - calitatea aerului, solului și a apei este recalculată

La începutul fiecărui test, veți primi **harta completă** împreună cu **toți parametrii inițiali**.

## Harta simulării

---



Imaginea de mai sus prezintă structura hărții și pozițiile exemplificative pentru TerraBot și elementele mediului.

Pentru detalii despre mișcările robotului, acțiuni și comenzi disponibile, consultați secțiunea Comenzi [<https://ocw.cs.pub.ro/courses/poo-ca-cd/teme/2025/8e211bfe-f6eb-467c-9e54-f8e6df4c1535/tema-1#comenzi>].

## Entități

---

Există mai multe tipuri de entități care se pot afla pe o pătrătică:

- Plant
- Animal
- Water
- Air
- Soil

## Caracteristici comune pentru toate entitățile

Toate entitățile de mai sus au următoarele atrbute comune:

Nume	Descriere	Tip
<b>name</b>	Numele entității	String
<b>mass</b>	În funcție de tipul clasei: - <b>soil</b> : greutatea totală a solului - <b>air</b> : greutatea tuturor gazelor găsite în acel tip de aer - <b>water</b> : greutatea apei lichide dintr-un rezervor (lac, râu, iaz). - <b>animal</b> : greutatea animalului. - <b>plant</b> : biomasa plantei.	double

## Plant

Tipurile de plante sunt împărțite în 5 categorii:

- FloweringPlants
- GymnospermsPlants
- Ferns
- Mosses
- Algae

## Interacțiuni cu mediul

O plantă are un singur tip de efect: ea crește nivelul de oxigen din aer, interacționând direct cu **mediul atmosferic**.

Fiecare plantă generează oxigen în atmosferă în funcție de:

- **Categoria plantei** → oferă oxigenul de bază
- **Nivelul de maturitate** → oferă un bonus

## Formula

```
oxigen_level += (maturity_oxygen_rate + oxigen_from_plant)
```

## Tabel combinat – Oxigen generat de plante

Categorie plantă	O <sub>2</sub> din categorie	Young (+0.2)	Mature (+0.7)	Old (+0.4)
FloweringPlants	6.0	6.2	6.7	6.4
GymnospermsPlants	0	0.2	0.7	0.4
Ferns	0	0.2	0.7	0.4
Mosses	0.8	1.0	1.5	1.2
Algae	0.5	0.7	1.2	0.9

De exemplu dacă avem:

- Air
  - oxigen\_level = 10
- Plant
  - name: Rose
  - type: FloweringPlants → oxigen\_from\_plant = 6
  - status: young → maturity\_oxygen\_rate = 0.2
- **Rezultat final: Oxigen\_level = 10 + 6 + 0.2 = 16.2**

Când nivelul de creștere devine **mai mare de o unitate (1.0)**, planta va trece la următorul nivel de maturitate:

- young → mature
- mature → old
- old → dead (**se va șterge planta de pe pătrătică**)

## Posibilitatea de a bloca TerraBot

Fiecare categorie de plantă are asociat un procentaj care indică **probabilitatea ca roboțelul să se agațe** de acea plantă.

## Formula

plant_possibility / 100.0
---------------------------

Probabilitate de agățare

Tip	Posibilitate
Flowering Plants (Angiosperms)	90%
Gymnosperms	60%
Ferns	30%
Mosses	40%
Algae	20%

## Animal

Animalele se deplasează pe hartă și consumă plantele întâlnite, ceea ce le permite să producă îngrășământ și să îmbunătățească calitatea solului.

Doar animalele scanate vor realiza acțiuni automate pe hartă, indiferent de proximitatea lor față de robot. Mai întâi se procesează hrănirea, apoi mișările.

### Mișcarea pe hartă

Animalul se mută pe una din pătrățelele vecine la fiecare 2 iterării pentru a se hrăni. Algoritmul de alegere este:

1. Se caută pătrățica care are plantă și apă. Dacă există mai multe, se alege cea cu calitatea apei mai bună.
2. Dacă nu există, se caută pătrățica care are plantă sau apă:
  - Dacă există plante, se alege prima pătrățică conform parcurgerii (sus, dreapta, jos, stânga).
  - Dacă nu există plante, se alege pătrățica cu apă, priorită de cea cu calitate mai bună.
  - Dacă nu există plante și apă, se alege prima pătrățică conform parcurgerii (sus, dreapta, jos, stânga).

### Starea animalului și producția de îngrășământ

Animalul poate fi în una din stările:

Stare	Condiție	Efect
hungry	Nu a mâncat sau nu a băut apă	Nu produce îngrășământ
well-fed	A mâncat Plant, Water sau Animal	Produce îngrășământ
sick	Aerul este toxic	Nu produce îngrășământ

Algoritmul de hrănire

#### 1. dacă animalul este de tip Carnivore sau Parasite

a. se va mâncă animalul de pe pătrățica curentă dacă există:

- se va crește masa **animalului curent**: **animalMass += preyMass**.
- la următoarea interacțiune **Animal** → **Soil**, entitatea **Soil** de pe pătrățica unde se află animalul la momentul interacțiunii va primi **+0.5 organicMatter**.
- **animalul mâncat** va dispărea de pe hartă.

b. dacă nu există un animal:

- dacă entitatea **Water** și entitatea **Plant** de pe pătrățică au fost deblocate deja la un pas anterior sau prezent de către robot:
  - **planta** va muri.
  - se va calcula **nivelul de apă de băut**: **waterToDrink = Math.min(animalMass \* intakeRate, waterMass)**, **intakeRate = 0.08**.
  - se va actualiza masa **apei**: **waterMass -= waterToDrink**.
  - se va crește masa **animalului curent**: **animalMass += (waterToDrink + plantMass)**.
  - la următoarea interacțiune **Animal** → **Soil**, entitatea **Soil** de pe pătrățica unde se află animalul la momentul interacțiunii va primi **+0.8 organicMatter**.

▪ se va mâncă entitatea **Water sau Plant**, în funcție de care entitate este deblocată prima de robot:

- dacă se va mâncă planta:
  - **planta** va muri.
  - se va crește masa **animalului curent**: **animalMass += plantMass**.
  - la următoarea interacțiune **Animal** → **Soil**, entitatea **Soil** de pe pătrățica unde se află animalul la momentul interacțiunii va primi **+0.5 organicMatter**.

▪ dacă se va bea apă:

- se va calcula **nivelul de apă de băut**: **waterToDrink = Math.min(animalMass \* intakeRate, waterMass)**, **intakeRate = 0.08**.
- se va actualiza masa **apei**: **waterMass -= waterToDrink**.
- se va crește masa **animalului curent**: **animalMass += waterToDrink**.
- la următoarea interacțiune **Animal** → **Soil**, entitatea **Soil** de pe pătrățica unde se află animalul la momentul interacțiunii va primi **+0.5 organicMatter**.

- dacă **nicio** entitate de pe pătrățica curentă nu a fost deblocată la un pas anterior sau prezent de către robot nu se va mâncă nimic.
- în această situație **nu** se va produce îngrășământ.

2. dacă animalul **nu** este de tip **Carnivore** sau **Parasite**:

- se reiau **toți pașii** descriși la pasul precedent pentru cazul "dacă nu există un animal".

Un animal consumă doar entitățile deblocate de robot.

Însă, animalele carnivore și parazite pot să se hrănească și cu animale nescanate.

### Posibilitatea de a ataca TerraBot

Animalele pot ataca robotul chiar dacă nu au fost deblocate.

#### Formulă

```
attack_probability = (100 - animal_possibility_to_attack) / 10.0
```

### Tabel de probabilități

Tip	Posibilitate
Herbivores	85%
Carnivores	30%
Omnivores	60%
Detritivores	90%
Parasites	10%

### Water

Apa este o resursă esențială în ecosistem, influențând direct solul, aerul, plantele și animalele. Fără apă, ciclurile biologice nu pot funcționa.

#### Caracteristici principale

Câmp	Descriere	Tip
salinity	Concentrația de sare a apei	double
pH	Aciditatea/alcalinitatea apei	double
purity	Nivelul de puritate al apei, în procente	double
turbidity	Nivelul de turbiditate al apei	double
contaminantIndex	Indice ce indică concentrația contaminanților	double
isFrozen	Indică dacă apa este înghețată	boolean

#### Interacțiuni cu mediul

Apa influențează direct celelalte entități ale ecosistemului:

Entitate	Efect	Rată / Iterație
Soil	Crește waterRetention	+0.1 timp de 2 iterări
Air	Crește humidity	+0.1 timp de 2 iterări
Plant	Contribuie la creștere	+0.2 la fiecare iterare

În cazul în care animalul consumă toata cantitatea din sursa de apă, atunci aceasta **va fi eliminată de pe hartă**.

#### Calculul calității apei

Calitatea apei se determină în doi pași:

1. Normalizarea factorilor:

```

purity_score      = purity / 100
pH_score          = 1 - abs(pH - 7.5) / 7.5
salinity_score    = 1 - (salinity / 350)
turbidity_score   = 1 - (turbidity / 100)
contaminant_score = 1 - (contaminantIndex / 100)
frozen_score       = isFrozen ? 0 : 1

```

2. Formula finală:

```

water_quality =
(0.3 * purity_score

```

```
+ 0.2 * pH_score
+ 0.15 * salinity_score
+ 0.1 * turbidity_score
+ 0.15 * contaminant_score
+ 0.2 * frozen_score) * 100
```

Interpretare rezultat

Interval	Calitate apă
70–100%	Good
40–70%	Moderate
< 40%	Poor

## Soil

Solul este esențial pentru creșterea plantelor și îmbunătățirea mediului. Acesta își poate crește calitatea prin absorția apei și prin primirea de îngrășământ.

### Caracteristici principale

Câmp	Descriere	Tip
nitrogen	Conținutul de azot (%)	double
waterRetention	Capacitatea solului de a reține apă	double
soilpH	Nivelul de aciditate/alcalinitate al solului	double
organicMatter	Conținutul de materie organică (%)	double

### Interacțiuni cu mediul

Solul interacționează cu plantele și apa:

Entitate	Efect	Rată / Iterație
Plant	Creștere +0.2	la fiecare iterare
Water	Absorbție +0.1 apă	la fiecare 2 iterări dacă există sursă de apă

### Calculul calității solului

Formula de calcul depinde de tipul solului:

Tip sol	Formula
ForestSoil	(nitrogen * 1.2) + (organicMatter * 2) + (waterRetention * 1.5) + (leafLitter * 0.3)
SwampSoil	(nitrogen * 1.1) + (organicMatter * 2.2) - (waterLogging * 5)
DesertSoil	(nitrogen * 0.5) + (waterRetention * 0.3) - (salinity * 2)
GrasslandSoil	(nitrogen * 1.3) + (organicMatter * 1.5) + (rootDensity * 0.8)
TundraSoil	(nitrogen * 0.7) + (organicMatter * 0.5) - (permafrostDepth * 1.5)

### Interpretare scor

Interval	Calitate sol
70–100%	Good
40–70%	Moderate
<40%	Poor

### Normalizarea scorului

```
normalizeScore = Math.max(0, Math.min(100, score))
```

### Rotunjirea scorului

```
finalResult = Math.round(normalizeScore * 100.0) / 100.0
```

**Calitatea solului** trebuie să fie **normalizată și rotunjită**.

### Probabilitate de a bloca TerraBot

Tip sol	Formulă probabilitate
ForestSoil	(waterRetention * 0.6 + leafLitter * 0.4) / 80 * 100
SwampSoil	waterLogging * 10
DesertSoil	(100 - waterRetention + salinity) / 100 * 100
GrasslandSoil	((50 - rootDensity) + waterRetention * 0.5) / 75 * 100
TundraSoil	(50 - permafrostDepth) / 50 * 100

## Air

Aerul este esențial pentru ecosistem, permitând animalelor să respire și să interacționeze cu mediul. Monitorizarea calității aerului este crucială pentru sănătatea animalelor și funcționarea TerraBot.

### Caracteristici principale

Câmp	Descriere	Tip
humidity	Umiditate relativă (%)	double
temperature	Temperatură ambientală (°C)	double
oxygenLevel	Concentrație de oxigen	double

### Calitatea aerului și toxicitatea

Calitatea aerului este calculată diferit pentru fiecare tip, folosind câmpurile specifice:

Tip de aer	Câmp specific	Formula calitate aer
tropical	co2Level	(oxygenLevel*2) + (humidity*0.5) - (co2Level*0.01)
polar	iceCrystalConcentration	(oxygenLevel*2) + (100-Math.abs(temperature)) - (iceCrystalConcentration*0.05)
temperat	pollenLevel	(oxygenLevel*2) + (humidity*0.7) - (pollenLevel*0.1)
de desert	dustParticles	(oxygenLevel*2) - (dustParticles*0.2) - (temperature*0.3)
montan	altitude	oxygenFactor = oxygenLevel - (altitude/1000*0.5) (oxygenFactor*2) + (humidity*0.6)

Interpretarea scorurilor calității aerului

Valoare	Interpretare
70-100%	good
40-70%	moderate
<40	poor

### Calculul toxicității aerului

Nivelul de toxicitate se calculează astfel:

```
toxicityAQ = 100 * (1 - airQualityScore / maxScore)
finalResult = Math.round(toxicityAQ * 100.0) / 100.0
```

Tip de aer	MaxScore
Tropical	82
Polar	142
Temperate	84
Desert	65
Mountain	78

Interpretarea scorurilor toxicității aerului

Dacă **toxicityAQ > (0.8 \* maxScore)**, atunci aerul este toxic.

### Normalizarea scorului

```
normalizeScore = Math.max(0, Math.min(100, score))
```

### Rotunjirea scorului

```
finalResult = Math.round(normalizeScore * 100.0) / 100.0
```

- **Normalizarea și rotunjirea se aplică pentru calculul calității și toxicității aerului.**
- **Calitatea aerului trebuie să fie normalizată și rotunjită înainte de a înlocui valoarea acesteia în formula pentru toxicitate.**

### Schimbări meteo

Evenimentele meteorologice modifică calitatea aerului:

Tip de aer	Eveniment	Formula actualizare
tropical	rainfall	normal_air_quality + (rainfall*0.3)
polar	polarStorm	normal_air_quality - (windSpeed*0.2)
temperat	newSeason	seasonPenalty = season.equalsIgnoreCase("Spring") ? 15 : 0 normal_air_quality - seasonPenalty
de desert	desertStorm	normal_air_quality - (desertStorm ? 30 : 0)
montan	peopleHiking	normal_air_quality - (numberOfHikers*0.1)

### Posibilitatea de a afecta TerraBot

Toxicitatea aerului și calitatea acestuia pot influența animalele și TerraBot. Se folosesc formulele din secțiunea Interacțiuni cu mediul pentru calcul.

## Interacțiuni între entități

În simulare, entitățile din mediu (Air, Soil, Water, Plant, Animal) pot interacționa între ele, influențându-se reciproc în timp.

Ordinea exactă în care sunt aplicate interacțiunile și actualizările este mereu aceeași:

1. Air
2. Soil
3. Water
4. Plant
5. Animal

**Toate interacțiunile cu Plant, Animal și Water sunt posibile doar după ce TerraBot a scanat entitatea respectivă.** Abia după scanare pot începe actualizările automate asociate acelei entități.

Dacă o entitate este eliminată de pe hartă, însă a fost scanată și salvată anterior în inventarul robotului, aceasta va rămâne în inventar în continuare.

### Frecvența interacțiunilor automate

Interacțiune	Efect	Frecvență
Air → Animal	Dacă aerul devine toxic → animalul devine sick	Verificat la fiecare iterație
Soil → Plant	+0.2 creștere la plantă	La fiecare iterație
Water → Air	+0.1 humidity	La fiecare 2 iterări
Water → Soil	+0.1 waterRetention	La fiecare 2 iterări
Water → Plant	+0.2 creștere la plantă	La fiecare iterăție
Plant → Air	+ oxigen în aer	La fiecare iterăție
Animal → Soil	+ organicMatter în sol dacă animalul devine well-fed	Verificat la fiecare iterăție
Animal → Water	+ animalMass la animal	La fiecare iterăție
Animal → Plant	Animalul mânâncă planta	Doar după ce planta a fost scanată

### Exemplu de întârziere a interacțiunilor

Dacă la **timestamp 3** se primește comanda `scanObject` pentru apă, atunci:

Abia la **timestamp 5** vor începe actualizările (conform frecvențelor de mai sus) pentru:

- Water → Soil
- Water → Air

Comanda de la timestamp 5 se execută doar după aplicarea tuturor actualizărilor automate.

### Reguli generale pentru interacțiuni

**Fiecare simulare este independentă.** Nu se păstrează inventarul robotului, energia sau starea mediului între simulări.

Simulările se execută **una după alta**, nu există simulări în paralel.

Comenzile trimise între `endSimulation` și următorul `startSimulation` sunt ignorate.

Fiecare celulă de pe hartă poate conține:

- **maxim 1** plantă
- **maxim 1** animal
- **maxim 1** sursă de apă

**Air** și **Soil** vor fi limitate tot doar la **o singură entitate** per tip, doar că acestea sunt complet disponibile de la începutul simulării, pe când restul entităților trebuie scanate înainte de a face anumite acțiuni cu alte entități.

## Comenzi

---

### Comenzi pentru simulare

#### `startSimulation`

- marchează începerea unei simulări
- robotul pornește **mereu** de pe **poziția [0, 0] a hărții**

Mesaje posibile pentru această comandă:

- "Simulation has started."

- "ERROR: Simulation already started. Cannot perform action"

**Input**

```
{
  "command": "startSimulation",
  "timestamp": 1
}
```

**Output**

```
{
  "command": "startSimulation",
  "message": "Simulation has started.",
  "timestamp": 1
}
```

**endSimulation**

- marchează sfârșitul unei simulări

Mesaje posibile pentru această comandă:

- "Simulation has ended."
- "ERROR: Simulation not started. Cannot perform action"

**Input**

```
{
  "command": "endSimulation",
  "timestamp": 10
}
```

**Output**

```
{
  "command": "endSimulation",
  "message": "Simulation has ended.",
  "timestamp": 10
}
```

**Comenzi pentru mediu****changeWeatherConditions**

- marchează un fenomen meteo care produce schimbări pentru aer (vezi secțiunea Air)
- Valorile posibile pentru tipul de schimbare meteo:
  - desertStorm → desert air
  - peopleHiking → mountain air
  - newSeason → temp air
  - polarStorm → polar air
  - rainfall → tropical air

Schimbarea meteo se ia în considerare **DOAR** pentru celulele care au acel tip de aer.

Când se primește comanda de changeWeatherConditions, **recalcularea calității aerului folosind formula inițială se va face peste 2 iteratii**.

**Exemplu**

- Se primește comanda changeWeatherConditions la **timestamp 3** pentru **desertStorm = true**.
- Se calculează calitatea aerului folosind formula pentru desertStorm
  - **normal\_air\_quality - (desertStorm ? 30 : 0)**
- Se salvează noua calitate a aerului
- Se primește comanda de la timestamp 4

- Se primește comanda de la **timestamp 5**
- Se recalculează calitatea aerului folosind **formula default** pentru *DesertAir*
  - $(\text{oxygenLevel} * 2) - (\text{dustParticles} * 0.2) - (\text{temperature} * 0.3)$

Mesaje posibile pentru această comandă:

- "The weather has changed."
- "ERROR: The weather change does not affect the environment. Cannot perform action"
- "ERROR: Simulation not started. Cannot perform action"

Input

```
{
  "command": "changeWeatherConditions",
  "type": "desertStorm",
  "desertStorm": true,
  "timestamp": 10
},
{
  "command": "changeWeatherConditions",
  "type": "rainfall",
  "rainfall": 12.5,
  "timestamp": 11
},
{
  "command": "changeWeatherConditions",
  "type": "peopleHiking",
  "numberOfHikers": 40,
  "timestamp": 12
}
```

Output

```
{
  "command": "changeWeatherConditions",
  "message": "The weather has changed.",
  "timestamp": 10
},
{
  "command": "changeWeatherConditions",
  "message": "The weather has changed.",
  "timestamp": 11
},
{
  "command": "changeWeatherConditions",
  "message": "The weather has changed.",
  "timestamp": 12
}
```

## Comenzi pentru TerraBot

### moveRobot

- **consumă energie**
- robotul se va muta pe pătrățelul cu scorul de calitate cel mai bun
- nu se poate explora mai departe de dimensiunea maximă a matricei
- se verifică dacă robotul are suficientă baterie pentru a se putea muta
- se poate trece în timpul simulării prin aceeași pătrățică
- consumul de baterie pentru mutare este raportat la scorul de calitate

Robotul nu poate să ramână pe pătrățica curentă. Dacă acesta mai are suficientă baterie pentru a efectua mutarea, atunci trebuie să aleagă neaparat una dintre pătrățelele vecine

### Deplasare robot

Se va duce pe pătrățelul cu scorul de calitate cel mai bun

- ordinea în care se verifică celulele: **sus, dreapta, jos, stânga**
- pentru calcularea scorului, se vor lua în considerare următoarele lucruri
  - **PossibilityToGetStuckInSoil**
    - pentru fiecare categorie de sol, se ia procentul de a rămâne blocat în el (secțiune Soil)
  - **PossibilityToGetDamagedByAir**

- pentru fiecare tip de aer, se iau valorile specifice și se înlocuiesc în formulele date (secțiune Air)
- **PossibilityToBeAttackedByAnimal**
  - pentru fiecare categorie de animale, se ia procentul de a fi atacat de ele (secțiune Animal)
- **PossibilityToGetStuckInPlants**
  - pentru fiecare categorie de plante, se ia procentul de a rămâne blocat în ele (secțiune Plant)
- **scorul final se calculează ca și medie aritmetică din cele 4 scoruri pentru fiecare posibilitate**
- i se va scădea din baterie numărul de puncte echivalent cu valoarea scorului obținut pentru celula pe care se va muta robotul (vezi exemplu mai jos)

Formulă scor calitate celulă

- ```

    double sum = sum(possibilityToGetStuckInSoil + possibilityToGetDamagedByAir + possibilityToBeAttackedByAnimal + possibilityToGetStuckInPlants)
    double mean = Math.abs(sum / count)
    int result = (int) Math.round(mean)

```

Dacă se obține același scor de calitate pentru una sau mai multe celule sau celula obținută este în afara hărții, atunci se alege prima din ordine în care s-au verificat celulele.

Se va verifica că robotul are suficientă baterie pentru a efectua mutarea.

#### Baterie robot

Robotul poate să rămână fără baterie înainte să se mute pe altă pătrătică

- dacă robotul nu mai are baterie **el nu se va deplasa până când nu se primește comanda rechargeBattery** pentru a-și încărca bateria, dar **restul pașilor se vor executa**, inclusiv cei de analiză și **funcționalitățile automate** din spate
- bateria se va consuma pentru fiecare mișcare a robotului pe altă celulă

#### Exemplu

Scor total de calitate pasul trecut = 2 → se vor scădea 2 unități din baterie

Mesaje posibile pentru această comandă:

- ```

    "The robot has successfully moved to position ${{coord_X, coord_Y}}."
    "ERROR: Not enough battery left. Cannot perform action"
    "ERROR: Robot still charging. Cannot perform action"
    "ERROR: Simulation not started. Cannot perform action"

```

#### Input

```
{
  "command": "moveRobot",
  "timestamp": 2
}
```

#### Output

```
{
  "command": "moveRobot",
  "message": "The robot has successfully moved to position (0,2)."
}
```

#### rechargeBattery

- se pună pauză simulării curente în derulare pentru ca roboțelul să se încarce unde se află în mod curent (are panouri solare 😎)
- se încarcă numărul de puncte de energie primite la input (**1 punct de energie = 1min de încărcare**)
  - exemplu: dacă se dă această **comandă la timestamp 5** pentru **20 de puncte de energie**, atunci următoarea comandă care poate fi **efectuată va fi la timestamp 25**

**Nu se poate pune pauză dacă o simulare tocmai s-a încheiat.**

Dacă nu s-a terminat încărcarea, se va returna eroare că roboțelul este încă la încărcat pentru orice comandă pentru robot dată în această perioadă.

Mesaje posibile pentru această comandă:

- "Robot battery is charging."
- "ERROR: Robot still charging. Cannot perform action"
- "ERROR: Simulation not started. Cannot perform action"

Input

```
{
  "command": "rechargeBattery",
  "timeToCharge": 20,
  "timestamp": 10
}
```

Output

```
{
  "command": "rechargeBattery",
  "message": "Robot battery is charging.",
  "timestamp": 10
}
```

scanObject

- **consumă energie**
- scanează un obiect (plantă, animal, apă) și salvează rezultatul (se salvează în baza de date și în inventar)
- se caută în harta salvată pentru secțiunea curentă de teritoriu dacă există un obiect de tipul respectiv pentru scanat (**nu este garantat că se găsește pe hartă**)
- consumă **7 puncte energie**

Roboțelul scanează obiectele de pe pătrățica curentă și determină tipul acestuia în funcție de 3 parametrii:

- color
- smell
- sound

Combinăriile de parametrii pentru obiecte:

- none + none + none = water
- pink + sweet + none = plant
- brown + earthy + muuu = animal

#### **! Atenție**

Câmpurile **color**, **smell** și **sound** pot avea și alte valori, nu doar cele enumerate mai sus (*pink, sweet, brown, earthy, muu*).

Se va verifica în baza de date că există pe pătrățică acel tip de obiect. Dacă da, se returnează restul de detalii pentru el și robotul va salva în baza de date și în inventarul sau noul obiect găsit. Dacă nu, se va returna eroare.

**Un obiect începe să existe abia după ce a fost scanat.** Așadar, o plantă/animal vor începe să își schimbe statusul doar după ce au fost scanate

Exemplu: un animal nu poate să mănânce planta de pe pătrățica curentă până aceasta nu a fost scanată.

Mesaje posibile pentru această comandă:

- "The scanned object is \${type\_of\_entity}"
- "ERROR: Object not found. Cannot perform action"
- "ERROR: Not enough energy to perform action"
- "ERROR: Robot still charging. Cannot perform action"
- "ERROR: Simulation not started. Cannot perform action"

Input

```
{
  "command": "scanObject",
  "color": "none",
  "smell": "none",
  "sound": "none",
  "timestamp": 4
},
{
  "command": "scanObject",
  "color": "pink",
  "smell": "sweet",
  "sound": "none",
  "timestamp": 5
},
{
  "command": "scanObject",
  "color": "brown",
  "smell": "earthy",
  "sound": "muu",
  "timestamp": 6
}
```

**Output**

```
{
  "command": "scanObject",
  "message": "The scanned object is water.",
  "timestamp": 4
},
{
  "command": "scanObject",
  "message": "The scanned object is a plant.",
  "timestamp": 5
},
{
  "command": "scanObject",
  "message": "The scanned object is an animal.",
  "timestamp": 6
}
```

**learnFact**

- **consumă energie**
- adaugă un nou fapt științific sub un anumit subiect
- se primește un subiect și componentele pentru acel subiect, mai multe detalii la comanda de *improveEnvironment*
- consumă **2 puncte energie**

**Exemplu**

Dacă TerraBot a descoperit (scanat) o plantă nouă **RedAlgae** și se primește comanda de *learnFact* ce include în **subiect** RedAlgae, va căuta în obiectele salvate dacă există această plantă. Dacă da, va salva informația dată sub numele / subiectul acestei plante.

**Nu se pot învăța facts până când TerraBot nu a scanat acel obiect**

Mesaje posibile pentru această comandă:

- "The fact has been successfully saved in the database."
- "ERROR: Subject not yet saved. Cannot perform action"
- "ERROR: Not enough battery left. Cannot perform action"
- "ERROR: Robot still charging. Cannot perform action"
- "ERROR: Simulation not started. Cannot perform action"

**Input**

```
{
  "command": "learnFact",
  "subject": "Method to plant RedAlgae",
  "components": "RedAlgae",
  "timestamp": 4
},
{
  "command": "learnFact",
  "subject": "Characteristic of RedAlgae",
  "timestamp": 5
}
```

```
{
  "components": "RedAlgae",
  "timestamp": 5
}
```

## Output

```
{
  "command": "learnFact",
  "message": "The fact has been successfully saved in the database.",
  "timestamp": 4
},
{
  "command": "learnFact",
  "message": "The fact has been successfully saved in the database.",
  "timestamp": 5
}
```

## improveEnvironment

- **consumă energie**
- robotul încearcă să îmbunătățească **mediul de pe pătrăică curentă** pentru a putea face planeta locuibilă
- **consumă 10 puncte energie**

Putem avea 4 tipuri de improvement:

- **plantVegetation**
  - +0.3 oxygen pentru aer față de cantitatea curentă
- **fertilizeSoil**
  - +0.3 organicMatter pentru sol
- **increaseHumidity**
  - +0.2 humidity la aer
- **increaseMoisture**
  - +0.2 waterRetention pentru sol

Ca să se efectueze comanda cu succes, trebuie să fie înndeplinite **2 condiții**:

- să se știe metoda prin care se realizează improvement-ul
  - trebuie să existe salvat **un fact de "Method to X"** ca să se poată face **improvementul X**
- robotul să aibă în inventar toate componentele necesare pentru improvement

*Hint: Practic e ca un fel de rețetă, trebuie să o cunoști și să ai ingredientele necesare ca să poți face improve.*

Mesaje posibile pentru această comandă:

- "The \${plant\_name} was planted successfully."
- "The \${improved\_entity} was successfully \${fertilized /increased } using \${component\_name}"
- "ERROR: Subject not yet saved. Cannot perform action"
- "ERROR: Fact not yet saved. Cannot perform action"
- "ERROR: Not enough battery left. Cannot perform action"
- "ERROR: Robot still charging. Cannot perform action"
- "ERROR: Simulation not started. Cannot perform action"

## Input

```
{
  "command": "improveEnvironment",
  "improvementType": "plantVegetation",
  "type": "Algae",
  "name": "RedAlgae",
  "timestamp": 6
},
{
  "command": "improveEnvironment",
  "improvementType": "fertilizeSoil",
  "type": "Soil",
  "name": "FertileSoil",
  "timestamp": 7
}
```

```
{
  "improvementType": "fertilizeSoil",
  "type": "Herbivores",
  "name": "Cow",
  "timestamp": 7
},
{
  "command": "improveEnvironment",
  "improvementType": "increaseHumidity",
  "type": "river",
  "name": "Blue River",
  "timestamp": 8
},
{
  "command": "improveEnvironment",
  "improvementType": "increaseMoisture",
  "type": "river",
  "name": "Blue River",
  "timestamp": 9
}
}
```

**Output**

```
{
  "command": "improveEnvironment",
  "message": "The RedAlgae was planted successfully.",
  "timestamp": 6
},
{
  "command": "improveEnvironment",
  "message": "The soil was successfully fertilized using Cow.",
  "timestamp": 7
},
{
  "command": "improveEnvironment",
  "message": "The humidity was successfully increased using Blue River.",
  "timestamp": 8
},
{
  "command": "improveEnvironment",
  "message": "The moisture was successfully increased using Blue River.",
  "timestamp": 9
}
```

**Comenzi de debug****getEnergyStatus**

- returnează energia curentă a TerraBot-ului

Mesaje posibile pentru această comandă:

- "TerraBot has \${energy\_points} energy points left."
- "ERROR: Simulation not started. Cannot perform action"

**Input**

```
{
  "command": "getEnergyStatus",
  "timestamp": 8
}
```

**Output**

```
{
  "command": "getEnergyStatus",
  "message": "TerraBot has 189 energy points left.",
  "timestamp": 8
}
```

**printEnvConditions**

- se printează pentru pătrățica curentă următoarele:
  - tipul de plantă, dacă există

- tipul de animal, dacă există
- tipul de sursă de apă, dacă există
- componenta solului
- componenta aerului

Se dorește monitorizarea furtunilor de nisip, astfel încât pentru tipul de aer **DesertAir**, va apărea în structura componentei aerului și **câmpul cu valori booleene "desertStorm"**, afectat de succesul comenzi **changeWeatherConditions**. În general, valoarea acestui câmp va fi **false**, dar în cazul declanșării furtunilor de nisip, pentru 2 iterații valoarea devine **true**.

Exemplu:

```
{
  "command": "printEnvConditions",
  "output": {
    ...
    "air": {
      "type": "DesertAir",
      "name": "Dry Desert Air",
      "mass": 900000.0,
      "humidity": 60.0,
      "temperature": 15.0,
      "oxygenLevel": 20.0,
      "airQuality": 3.0,
      "desertStorm": true
    }
  },
  "timestamp": 22
},
```

Mesaje posibile pentru această comandă:

- **"ERROR: Simulation not started. Cannot perform action"**

Input

```
{
  "command": "printEnvConditions",
  "timestamp": 5
}
```

Output

```
{
  "command": "printEnvConditions",
  "output": {
    "soil": {
      "type": "ForestSoil",
      "name": "LightForestSoil",
      "mass": 1000.0,
      "nitrogen": 2.5,
      "waterRetention": 0.6,
      "soilph": 6.7,
      "organicMatter": 0.3,
      "soilQuality": 4.56,
      "leafLitter": 0.2
    },
    "plants": {
      "type": "FloweringPlants",
      "name": "Tulip",
      "mass": 0.5
    },
    "animals": {
      "type": "Herbivores",
      "name": "Rabbit",
      "mass": 2.66
    },
    "water": {
      "type": "pond",
      "name": "SmallPond",
      "mass": 29.84,
      "purity": "85.0",
      "salinity": "0.1",
      "turbidity": 2.0,
      "contaminantIndex": 0.5,
      "temperature": 15.0,
      "pH": 6.6,
      "isFrozen": false,
      "volumeLiters": 100.0
    },
    "air": {
      "type": "TemperateAir",
      "name": "Spring Air",
      "mass": 900000.0,
      "humidity": 60.1,
      "temperature": 15.0,
      "oxygenLevel": 26.2,
      "desertStorm": false
    }
  },
  "timestamp": 5
}
```

```

        "airQuality" : 93.47,
        "pollenLevel" : 10.0
    }
},
"timestamp" : 5
}

```

**printMap**

- se printează pentru fiecare pătrătică de pe hartă următoarele:
  - numărul total de enități de pe pătrătică (sum(plant, animal, water\_source))
  - calitate aerului
  - calitatea solului

Mesaje posibile pentru această comandă:

- **"ERROR: Simulation not started. Cannot perform action"**

**Input**

```
{
  "command": "printMap",
  "timestamp": 23
}
```

**Output**

```
{
  "command": "printMap",
  "output": [
    {
      "section": [0,0],
      "totalNrOfObjects": 1,
      "airQuality": "good",
      "soilQuality": "good"
    },
    {
      "section": [0,1],
      "totalNrOfObjects": 0,
      "airQuality": "good",
      "soilQuality": "good"
    },
    {
      "section": [0,2],
      "totalNrOfObjects": 2,
      "airQuality": "moderate",
      "soilQuality": "good"
    },
    {
      "section": [1,0],
      "totalNrOfObjects": 0,
      "airQuality": "poor",
      "soilQuality": "poor"
    },
    {
      "section": [1,1],
      "totalNrOfObjects": 3,
      "airQuality": "good",
      "soilQuality": "good"
    },
    {
      "section": [1,2],
      "totalNrOfObjects": 1,
      "airQuality": "moderate",
      "soilQuality": "poor"
    },
    {
      "section": [2,0],
      "totalNrOfObjects": 2,
      "airQuality": "good",
      "soilQuality": "moderate"
    },
    {
      "section": [2,1],
      "totalNrOfObjects": 3,
      "airQuality": "moderate",
      "soilQuality": "good"
    },
    {
      "section": [2,2],
      "totalNrOfObjects": 0,
      "airQuality": "poor",
      "soilQuality": "poor"
    }
  ]
}
```

```
{
    "timestamp": 23
}
```

### printKnowledgeBase

- afisează pentru fiecare tip de entitate fact-urile învățate
  - exemplu: s-a scanat și învățat despre RedAlgae, se printează toate facts despre RedAlgae
- ordinea de printare va fi:
  - topic-urile vor fi în ordinea în care s-a efectuat scanarea lor
  - facts vor fi în ordinea în care au fost învățate de robot

Mesaje posibile pentru această comandă:

- "ERROR: Simulation not started. Cannot perform action"

### Input

```
{
    "command": "printKnowledgeBase",
    "timestamp": 17
}
```

### Output

```
{
    "command": "printKnowledgeBase",
    "output": [
        {
            "topic": "RedAlgae",
            "facts": [ "Method to plant RedAlgae", "Characteristic of RedAlgae" ]
        }
    ],
    "timestamp": 17
}
```

---

## Exemplu simulare

Pentru a ilustra mai bine ordinea și efectul operațiilor din secțiunile anterioare, vă atașăm un exemplu minimal al unei simulări.

### Input

```
{
    "simulationParams": [
        {
            "territoryDim": "3x3",
            "energyPoints": 22,
            "territorySectionParams": {
                "soil": [
                    {
                        "type": "ForestSoil",
                        "name": "LightForestSoil",
                        "mass": 1000.0,
                        "nitrogen": 2.5,
                        "waterRetention": 0.5,
                        "soilpH": 6.7,
                        "organicMatter": 0.3,
                        "leafLitter": 0.2,
                        "sections": [
                            {
                                "x": 0,
                                "y": 0
                            },
                            {
                                "x": 0,
                                "y": 1
                            },
                            {
                                "x": 0,
                                "y": 2
                            }
                        ]
                    },
                    {
                        "type": "SwampSoil",

```

```

        "name": "WetPeat",
        "mass": 1100.0,
        "nitrogen": 2.0,
        "waterRetention": 0.7,
        "soilpH": 6.3,
        "organicMatter": 0.5,
        "waterLogging": 0.6,
        "sections": [
            {
                "x": 1,
                "y": 0
            },
            {
                "x": 1,
                "y": 1
            },
            {
                "x": 1,
                "y": 2
            }
        ]
    },
    {
        "type": "DesertSoil",
        "name": "DrySand",
        "mass": 800.0,
        "nitrogen": 0.8,
        "waterRetention": 0.15,
        "soilpH": 7.2,
        "organicMatter": 0.1,
        "salinity": 0.05,
        "sections": [
            {
                "x": 2,
                "y": 0
            },
            {
                "x": 2,
                "y": 1
            },
            {
                "x": 2,
                "y": 2
            }
        ]
    }
],
"plants": [
    {
        "type": "FloweringPlants",
        "name": "Tulip",
        "mass": 0.5,
        "sections": [
            {
                "x": 0,
                "y": 0
            },
            {
                "x": 1,
                "y": 1
            },
            {
                "x": 2,
                "y": 2
            },
            {
                "x": 0,
                "y": 1
            }
        ]
    }
],
"animals": [
    {
        "type": "Herbivores",
        "name": "Rabbit",
        "mass": 2,
        "sections": [
            {
                "x": 1,
                "y": 0
            },
            {
                "x": 2,
                "y": 1
            },
            {
                "x": 0,
                "y": 1
            }
        ]
    }
],
"water": [
    {
        "type": "pond",
        "name": "SmallPond",
        "mass": 30.0,
        "sections": [
            {
                "x": 0,
                "y": 0
            }
        ]
    }
]
}

```

```

        "purity": 85.0,
        "salinity": 0.1,
        "turbidity": 2,
        "contaminantIndex": 0.5,
        "pH": 6.6,
        "isFrozen": false,
        "sections": [
            {
                "x": 0,
                "y": 2
            },
            {
                "x": 1,
                "y": 2
            },
            {
                "x": 0,
                "y": 1
            },
            {
                "x": 1,
                "y": 0
            }
        ]
    },
    "air": [
        {
            "type": "TemperateAir",
            "name": "Spring Air",
            "mass": 900000.0,
            "humidity": 60.0,
            "temperature": 15.0,
            "oxygenLevel": 20,
            "pollenLevel": 10.0,
            "sections": [
                {
                    "x": 0,
                    "y": 0
                },
                {
                    "x": 1,
                    "y": 1
                },
                {
                    "x": 2,
                    "y": 2
                },
                {
                    "x": 0,
                    "y": 1
                },
                {
                    "x": 1,
                    "y": 0
                },
                {
                    "x": 2,
                    "y": 1
                },
                {
                    "x": 0,
                    "y": 2
                },
                {
                    "x": 1,
                    "y": 2
                },
                {
                    "x": 2,
                    "y": 0
                }
            ]
        }
    ]
},
"commands": [
    {
        "command": "startSimulation",
        "timestamp": 1
    },
    {
        "command": "moveRobot",
        "timestamp": 2
    },
    {
        "command": "scanObject",
        "color": "pink",
        "smell": "sweet",
        "sound": "none",
        "timestamp": 3
    },
    {
        "command": "printEnvConditions",
        "timestamp": 4
    }
],

```

```
{
    "command": "learnFact",
    "subject": "Method to plant Tulip",
    "components": "Tulip",
    "timestamp": 5
},
{
    "command": "improveEnvironment",
    "improvementType": "plantVegetation",
    "type": "FloweringPlants",
    "name": "Tulip",
    "timestamp": 6
},
{
    "command": "printEnvConditions",
    "timestamp": 7
},
{
    "command": "getEnergyStatus",
    "timestamp": 8
},
{
    "command": "printMap",
    "timestamp": 9
},
{
    "command": "endSimulation",
    "timestamp": 10
},
{
    "command": "moveRobot",
    "timestamp": 11
}
]
```

Schimbarea datelor pentru hartă constă în actualizarea parametrilor pentru mediu - **air, soil, water (în această ordine)** - și apoi plantele și animalele (**în această ordine**)

Recomandăm rotunjirea fiecărui rezultat final de tip **double** înainte de a actualiza câmpurile unui obiect cu acesta. Puteți folosi formula de mai jos:

```
Math.round(result * 100.0) / 100.0
```

## Desfășurarea simulării

- Se citesc datele de la input și se reține preambulul cu parametrii pentru simulare.
- Se începe prelucrarea comenzilor

### startSimulation

- Se încarcă datele pentru simularea curentă
- Se începe simularea de pe poziția (0, 0).
- Încă nu s-au scanat obiecte, deci nu se vor face schimbări automate pentru mediu sau entități.

### moveRobot

- Se fac actualizările automate pentru interacțiuni, plante și animale
  - Încă nu s-a efectuat o scanare, aşadar încă nu avem interacțiuni
- Se fac actualizările automate pentru mediu - sol și aer
  - Se calculează doar calitatea aerului și a solului, întrucât robotul încă nu a scanat o entitate
  - Avem TemperateAir → se aplică formula specifică acestui tip de aer
    - **double score = (oxygenLevel \* 2) + (humidity \* 0.7) - (pollenLevel \* 0.1) = 81.0**
    - **double airQuality = Math.round(score \* 100.0) / 100.0 = 81.0**
  - Avem ForestSoil → se aplică formula specifică acestui tip de sol
    - **double score = (nitrogen \* 1.2) + (organicMatter \* 2) + (waterRetention \* 1.5) + (leafLitter \* 0.3) = 4.409999999999999**
    - **double soilQuality = Math.round(score \* 100.0) / 100.0 = 4.41**
- Se identifică pătrățelele vecine: (0, 1) și (1, 0)
- Se parcurg pătrățelele vecine în ordinea sus, dreapta, jos, stânga ⇒ se va parurge (0, 1) și apoi (1, 0)
- Celula (0, 1) are: animal, ForestSoil, apă, TemperateAir, plantă
- Celula (1, 0) are: animal, SwampSoil, apă, TemperateAir
- Se calculează fiecare posibilitate de daune pentru robot:

Category	Celula (1, 0)	Celula (0, 1)
PossibilityToGetStuckInSoil	6.0	0.475
PossibilityToGetDamagedByAir	3.57	3.57
PossibilityToBeAttackedByAnimal	1.5	1.5
PossibilityToGetStuckInPlants	0	0.9

- Scorul final pentru pătrățica (1, 0):  $\text{Math.round}(3.69) = 4$
- Scorul final pentru pătrățica (0, 1):  $\text{Math.round}(1.61125) = 2$
- Scorul pentru pătrățica (0, 1) este mai bun  $\Rightarrow$  robotul se mută pe pătrățica (0, 1)
  - Un scor mai mic presupune șanse mai scăzute ca robotul să primească daune
- Se verifică dacă robotul mai are suficientă baterie pentru a se muta
  - nu s-au efectuat alte comenzi care să consume baterie, aşadar robotul are 22 puncte energie
  - încă are suficientă baterie, deci se va afișa mesaj de succes
- Se actualizează cantitatea de baterie necesară pentru mutare:  $22 - 2 = 20$

**scanObject**

- Se fac actualizările automate pentru interacțiuni, plante și animale
  - încă nu s-a efectuat o scanare, aşadar încă nu avem interacțiuni
- Se fac actualizările automate pentru mediu - sol și aer
  - Întrucât tipul de soil și air sunt de același tip ca și celula anterioară, vom avea aceleași valori pentru calitatea solului și a aerului  $\rightarrow \text{airQuality} = 81.0$  și  $\text{soilQuality} = 4.41$
- Se verifică dacă robotul are minim 7 puncte de energie rămase pentru a efectua comanda
  - are 18 puncte ramase  $\rightarrow$  se va efectua comanda
- Se actualizează cantitatea de baterie:  $20 - 7 = 13$
- Robotul scană un obiect
- Pe baza celor 3 parametrii scanati se determină că obiectul scanat este o plantă (pink + sweet + none)
- Se caută în baza de date a simulării dacă există plantă pe celula curentă (0, 1)
  - se găsește că există o plantă Tulip și se returnează restul de detalii pentru plantă (type, name, etc.)
- Se salvează plantă Tulip în inventar

**printEnvConditions**

- Se fac actualizările automate pentru interacțiuni, plante și animale
  - interacțiuni
    - Soil  $\rightarrow$  Plant
      - planta curentă este Tulip și abia a fost scanată  $\rightarrow$  va avea status "young" și growth rate 0
      - solul ajută plantă să crească cu 0.2  $\Rightarrow$  growthRate =  $0 + 0.2 = 0.2$
    - Plant  $\rightarrow$  Air
      - aerul este de tip TemperateAir și are oxygenLevel = 20 (este valoarea de la citire nemodificată)
      - planta are statusul "young"  $\rightarrow$  improveRate = 0.2
      - planta este de tipul FloweringPlants  $\rightarrow$  oxygenFromPlant = 6.0
      - se va actualiza nivelul de oxigen: oxygenLevel =  $20 + 6.0 + 0.2 = 26.2$
- Se fac actualizările automate pentru mediu - sol și aer
  - noile valori pentru mediu: airQuality = 93.4 și soilQuality = 4.41
  - Întrucât doar aerul a suferit modificări, doar calitatea acestuia se va actualiza, iar pentru sol va rămâne la fel
- Se printeză toate detaliile despre pătrățica curentă

**learnFact**

- Se fac actualizările automate pentru interacțiuni, plante și animale
  - interacțiuni
    - Soil  $\rightarrow$  Plant
      - solul ajută plantă să crească cu 0.2  $\Rightarrow$  growthRate =  $0.2 + 0.2 = 0.4$
    - Plant  $\rightarrow$  Air
      - oxygenLevel = 26.2
      - planta are statusul "young"  $\rightarrow$  improveRate = 0.2
      - planta este de tipul FloweringPlants  $\rightarrow$  oxygenFromPlant = 6.0
      - se va actualiza nivelul de oxigen: oxygenLevel =  $26.2 + 6.0 + 0.2 = 32.4$
- Se fac actualizările automate pentru mediu - sol și aer
  - valorile pentru mediu (se modifică doar pentru aer): airQuality = 100.0 și soilQuality = 4.41
- Se verifică dacă robotul are minim 2 puncte de energie rămase pentru a efectua comanda
- Se actualizează cantitatea de baterie:  $13 - 2 = 11$
- Se verifică dacă obiectul pentru care s-a dat fact-ul există în inventarul robotului (ce este în câmpul de "components")
  - există plantă Tulip în inventar  $\rightarrow$  se poate salva fact-ul
- Fact-ul se va salva în baza de date a robotului sub obiectul Tulip

**improveEnvConditions**

- Se fac actualizările automate pentru interacțiuni, plante și animale
  - interacțiuni
    - Soil  $\rightarrow$  Plant
      - solul ajută plantă să crească cu 0.2  $\Rightarrow$  growthRate =  $0.4 + 0.2 = 0.6$

- Plant → Air
  - oxygenLevel = 32.4
  - planta are statusul "young" → improveRate = 0.2
  - planta este de tipul FloweringPlants → oxygenFromPlant = 6.0
  - se va actualiza nivelul de oxigen: oxygenLevel =  $32.4 + 6.0 + 0.2 = 38.6$
- Se fac actualizările automate pentru mediu - sol si aer
  - valorile pentru mediu (nu se modifică): airQuality = 100.0 și soilQuality = 4.41
  - valoarea maximă pentru calitate este 100.0 → calitatea aerului nu se va actualiza
- Se verifică dacă robotul are minim 10 puncte de energie rămase pentru a efectua comanda
  - De la comanda precedentă mai avem 11 puncte de energie, aşadar se va efectua comanda
- \* Se actualizează cantitatea de baterie:  $11 - 10 = 1$
- Se verifică cele 2 condiții pentru comandă:
  - dacă există în inventar obiectul cu nume "Tulip"
  - dacă pentru obiectul de tip "Tulip" s-a salvat fact de tipul "method for "Method to plant Tulip"
- Cum cele 2 condiții sunt îndeplinite, se va aplica improvement-ul pe pătrățică
  - aerul este de tip TemperateAir și are oxygenLevel = 38.6
  - tipul de improvement este "plantVegetation" → trebuie actualizat oxygenLevel cu + 0.3
  - oxygenLevel =  $38.6 + 0.3 = 38.9$
  - se elimină din inventarul robotului "Tulip"
  - se va printa mesajul de succes pentru comandă

**printEnvConditions**

- Se fac actualizările automate pentru interacțiuni, plante și animale
  - interacțiuni
    - Soil → Plant
      - solul ajută planta să crească cu 0.2 ⇒ growthRate =  $0.6 + 0.2 = 0.8$
    - Plant → Air
      - oxygenLevel = 38.6
      - planta are statusul "young" → improveRate = 0.2
      - planta este de tipul FloweringPlants → oxygenFromPlant = 6.0
      - se va actualiza nivelul de oxigen: oxygenLevel =  $38.6 + 6.0 + 0.2 = 45.1$
- Se fac actualizările automate pentru mediu - sol si aer
  - valorile pentru mediu (nu se modifica): airQuality = 100.0 și soilQuality = 4.41
- Se printează toate detaliile despre pătrățica curentă

**getEnergyStatus**

- Se fac actualizările automate pentru interacțiuni, plante și animale
  - interacțiuni
    - Soil → Plant
      - solul ajută planta să crească cu 0.2 ⇒ growthRate =  $0.8 + 0.2 = 1.0$
      - s-a ajuns la growthRate = 1.0, deci trebuie schimbat statusul pentru plantă → din "young" va deveni "mature"
      - se va reseta growthRate la 0
    - Plant → Air
      - oxygenLevel = 45.1
      - planta are statusul "mature" → improveRate = 0.7
      - planta este de tipul FloweringPlants → oxygenFromPlant = 6.0
      - se va actualiza nivelul de oxigen: oxygenLevel =  $45.1 + 6.0 + 0.7 = 51.8$
- Se fac actualizările automate pentru mediu - sol și aer
  - valorile pentru mediu (nu se modifica): airQuality = 100.0 și soilQuality = 4.41
- Se printează bateria rămasă a robotului la momentul curent
  - mai este doar 1 punct de energie rămas în baterie

**printMap**

- Se fac actualizările automate pentru interacțiuni, plante și animale
  - interacțiuni
    - Soil → Plant
      - solul ajută planta să crească cu 0.2 ⇒ growthRate =  $0 + 0.2 = 0.2$
    - Plant → Air
      - oxygenLevel = 51.8
      - planta are statusul "mature" → improveRate = 0.7
      - planta este de tipul FloweringPlants → oxygenFromPlant = 6.0
      - se va actualiza nivelul de oxigen: oxygenLevel =  $51.8 + 6.0 + 0.7 = 58.5$
- Se fac actualizările automate pentru mediu - sol și aer

- valorile pentru mediu (nu se modifica): airQuality = 100.0 și soilQuality = 4.41
- Se printează pentru fiecare pătrăpică numărul de entități optionale - plantă, animal, apă - și calitatea solului și a aerului până la momentul curent

**endSimulation**

- Se încheie simularea
- Nu se mai pot procesa alte comenzi
- Se resetează câmpurile pentru a începe o nouă simulare

**moveRobot**

- comanda aceasta este primită după încheierea simulării, prin urmare nu se va efectua și se va afișa mesajul de eroare aferent

**Output**

```
[
  {
    "command": "startSimulation",
    "message": "Simulation has started.",
    "timestamp": 1
  },
  {
    "command": "moveRobot",
    "message": "The robot has successfully moved to position (0, 1).",
    "timestamp": 2
  },
  {
    "command": "scanObject",
    "message": "The scanned object is a plant.",
    "timestamp": 3
  },
  {
    "command": "printEnvConditions",
    "output": {
      "soil": {
        "type": "ForestSoil",
        "name": "LightForestSoil",
        "mass": 1000.0,
        "nitrogen": 2.5,
        "waterRetention": 0.5,
        "soilpH": 6.7,
        "organicMatter": 0.3,
        "soilQuality": 4.41,
        "leafLitter": 0.2
      },
      "plants": {
        "type": "FloweringPlants",
        "name": "Tulip",
        "mass": 0.5
      },
      "animals": {
        "type": "Herbivores",
        "name": "Rabbit",
        "mass": 2.0
      },
      "water": {
        "type": "pond",
        "name": "SmallPond",
        "mass": 30.0,
        "purity": 85.0,
        "salinity": 0.1,
        "turbidity": 2.0,
        "contaminantIndex": 0.5,
        "temperature": 15.0,
        "ph": 6.6,
        "isFrozen": false
      },
      "air": {
        "type": "TemperateAir",
        "name": "Spring Air",
        "mass": 900000.0,
        "humidity": 60.0,
        "temperature": 15.0,
        "oxygenLevel": 26.2,
        "airQuality": 93.4,
        "pollenLevel": 10.0
      }
    },
    "timestamp": 4
  },
  {
    "command": "learnFact",
    "message": "The fact has been successfully saved in the database.",
    "timestamp": 5
  },
  {
    "command": "improveEnvironment",
    "message": "The Tulip was planted successfully.",
    "timestamp": 6
  },
  {
    "command": "printEnvConditions",
    "output": {
      "soil": {
        "type": "ForestSoil",
        "name": "LightForestSoil",
        "mass": 1000.0,
        "nitrogen": 2.5,
        "waterRetention": 0.5,
        "soilpH": 6.7,
        "organicMatter": 0.3,
        "soilQuality": 4.41,
        "leafLitter": 0.2
      },
      "plants": {
        "type": "FloweringPlants",
        "name": "Tulip",
        "mass": 0.5
      },
      "animals": {
        "type": "Herbivores",
        "name": "Rabbit",
        "mass": 2.0
      },
      "water": {
        "type": "pond",
        "name": "SmallPond",
        "mass": 30.0,
        "purity": 85.0,
        "salinity": 0.1,
        "turbidity": 2.0,
        "contaminantIndex": 0.5,
        "temperature": 15.0,
        "ph": 6.6,
        "isFrozen": false
      },
      "air": {
        "type": "TemperateAir",
        "name": "Spring Air",
        "mass": 900000.0,
        "humidity": 60.0,
        "temperature": 15.0,
        "oxygenLevel": 26.2,
        "airQuality": 93.4,
        "pollenLevel": 10.0
      }
    },
    "timestamp": 7
  }
]
```

```

    "output": {
      "soil": {
        "type": "ForestSoil",
        "name": "LightForestSoil",
        "mass": 1000.0,
        "nitrogen": 2.5,
        "waterRetention": 0.5,
        "soilpH": 6.7,
        "organicMatter": 0.3,
        "soilQuality": 4.41,
        "leafLitter": 0.2
      },
      "plants": {
        "type": "FloweringPlants",
        "name": "Tulip",
        "mass": 0.5
      },
      "animals": {
        "type": "Herbivores",
        "name": "Rabbit",
        "mass": 2.0
      },
      "water": {
        "type": "pond",
        "name": "SmallPond",
        "mass": 30.0,
        "purity": 85.0,
        "salinity": 0.1,
        "turbidity": 2.0,
        "contaminantIndex": 0.5,
        "temperature": 15.0,
        "pH": 6.6,
        "isFrozen": false
      },
      "air": {
        "type": "TemperateAir",
        "name": "Spring Air",
        "mass": 900000.0,
        "humidity": 60.0,
        "temperature": 15.0,
        "oxygenLevel": 45.1,
        "airQuality": 100.0,
        "pollenLevel": 10.0
      }
    },
    "timestamp": 7
  },
  {
    "command": "getEnergyStatus",
    "message": "TerraBot has 1 energy points left.",
    "timestamp": 8
  },
  {
    "command": "printMap",
    "output": [
      {
        "section": [
          0,
          0
        ],
        "totalNrOfObjects": 1,
        "airQuality": "good",
        "soilQuality": "poor"
      },
      {
        "section": [
          1,
          0
        ],
        "totalNrOfObjects": 2,
        "airQuality": "good",
        "soilQuality": "poor"
      },
      {
        "section": [
          2,
          0
        ],
        "totalNrOfObjects": 0,
        "airQuality": "good",
        "soilQuality": "poor"
      },
      {
        "section": [
          0,
          1
        ],
        "totalNrOfObjects": 3,
        "airQuality": "good",
        "soilQuality": "poor"
      },
      {
        "section": [
          1,
          1
        ],
        "totalNrOfObjects": 1,
        "airQuality": "good",
        "soilQuality": "poor"
      }
    ]
  }
]

```

```

        "soilQuality": "poor"
    },
    {
        "section": [
            2,
            1
        ],
        "totalNrOfObjects": 1,
        "airQuality": "good",
        "soilQuality": "poor"
    },
    {
        "section": [
            0,
            2
        ],
        "totalNrOfObjects": 1,
        "airQuality": "good",
        "soilQuality": "poor"
    },
    {
        "section": [
            1,
            2
        ],
        "totalNrOfObjects": 1,
        "airQuality": "good",
        "soilQuality": "poor"
    },
    {
        "section": [
            2,
            2
        ],
        "totalNrOfObjects": 1,
        "airQuality": "good",
        "soilQuality": "poor"
    }
],
"timestamp": 9
},
{
    "command": "endSimulation",
    "message": "Simulation has ended.",
    "timestamp": 10
},
{
    "command": "moveRobot",
    "message": "ERROR: Simulation not started. Cannot perform action",
    "timestamp": 11
}
]

```

## Scheletul de cod

În rezolvarea temei va fi nevoie de folosirea unor obiecte pentru stocarea și maparea datelor primite în format JSON. Scheletul temei vă oferă mai multe clase utile pentru citirea și rularea temei. Acestea se regăsesc în cadrul scheletului, astfel:

- Clasele de input din cadrul pachetului **fileio**: Acestea se vor folosi pentru parsarea datelor de input. Astfel, preluăți toate informațiile necesare pentru a crea propriile clase pentru rezolvarea temei.
- Clasa **Main** care este punctul de intrare pentru logica de rezolvare a temei.

Pentru a înțelege mai bine cum funcționează citirea/scrierea în fișierele JSON vă recomandăm să citiți Json & Jackson [<https://ocw.cs.pub.ro/courses/poo-ca-cd/laboratoare/tutorial-json-jackson>] sau Jackson Object Mapper tutorial (Baeldung) [<https://www.baeldung.com/jackson-object-mapper-tutorial>].

- Implementarea voastră va începe din metoda **Main.action**, unde veți adăuga în **ArrayNode output** pe parcursul execuției toate output-urile comezilor date. **De asemenea, aveți un mic exemplu în schelet despre cum puteți face asta.**
- Aveți în fișierul **pom.xml** toate dependințele necesare pentru rularea temei, mai exact bibliotecile Jackson și câteva dependințe folosite pentru testare.

Output-ul **nu trebuie** formatat ca în ref-uri, fiindcă se verifică conținutul obiectelor și array-urilor JSON, nu textul efectiv.

Totuși vă recomandăm să utilizați **PrettyPrinter** dacă folosiți Jackson pentru că vă ajută la depanare ( Documentație PrettyPrinter [[https://fasterxml.github.io/jackson-databind/javadoc/2.7/com/fasterxml/jackson/databind/ObjectMapper.html#writeWithDefaultPrettyPrinter\(\)](https://fasterxml.github.io/jackson-databind/javadoc/2.7/com/fasterxml/jackson/databind/ObjectMapper.html#writeWithDefaultPrettyPrinter())]).

Totodată, pentru a înțelege cum se poate realiza **scrierea în fișierele JSON de output**, vă sugerăm să consultați JSON Array [<https://attacomsian.com/blog/jackson-create-json-array>].

Dacă aveți probleme cu proiectul vostru **vă recomandăm** următoarele soluții:

- Să închideți proiectul din IntelliJ, să ștergeți din file explorer folderul .idea și target și să deschideți din nou folderul proiectului cu IntelliJ (File → open → selectați folderul în care se află pom.xml). Acest pas vă **regenerează configurația IntelliJ-ului**.
- Să apăsați pe iconita "M" din dreapta în IntelliJ, după care să apăsați pe item-ul din listă, apoi pe lifecycle și apoi dublu click pe install. Acest pas vă **reinstalează dependințele proiectului**.
- Să apăsați pe File → Invalidate cache și să bifați toate opțiunile. Proiectul o să se reîncarce **ștergând toate configurațiile și fișierele cached**.

## Execuția temei

---

Pentru a rula checker-ul intrați în fișierul “`src/test/java/TestRunner`” și rulați clasa.

O să vi se deschidă un panou în IntelliJ de unde puteți vedea **statusul tuturor testelor**. De asemenea, vi se va afișa și **eroarea sau diferența rezultatului** pe care îl aveți pentru **fiecare test** în parte.

Pentru ca checker-ul să funcționeze trebuie să deschideți tema din IntelliJ la calea unde se află fișierul “`pom.xml`”.

## Recomandări

---

- Tema a fost concepută pentru a vă testa cunoștințele dobândite în urma parcurgerii laboratoarelor 1-6, aceasta putând fi rezolvată doar cu noțiunile invățate din acele laboratoare.
- Tema fiind o specificație destul de stufoasă recomandăm citirea de cel puțin două ori a enunțului înainte de inceperea implementării.

**Verificați periodic această pagină**, deoarece scheletul/cerința pot suferi modificări în urma unor erori din partea noastră.

## Utilizarea LLM-urilor (ChatGPT, Gemini, Claude etc.)

Puteți folosi modele de limbaj (LLM-uri) pentru **asistență** în realizarea temei doar ca **instrument suplimentar**, de exemplu pentru clarificări, explicații sau verificarea ideilor.

Totuși, **recomandarea noastră** este să încercați să **rezolvați tema pe cont propriu**, fără a depinde de astfel de instrumente. Scopul exercițiului este să **învățați** și să vă **dezvoltăți abilitățile**, iar progresul real vine doar prin înțelegere și practică personală.

Puteți folosi LLM-uri pentru:

- explicații teoretice.
- exemple suplimentare.
- verificarea codului sau a ideilor.
- clarificarea unor concepte.

**NU** este acceptat să:

- trimiteți soluții generate integral de un LLM.
- folosiți cod sau text pe care nu îl înțelegeți.
- copiați fără să puteți explica ce ați făcut în cod.
- generați README-ul folosind un LLM.

## Evaluare

---

Punctajul constă din:

- 80p implementare - trecerea testelor
- 10p coding style (vezi checkstyle)
- 5p README.md clar, concis, explicații axate pe design (flow, interacțiuni)
- 5p folosire git pentru versionarea temei
- Pe pagina Indicații pentru teme găsiți indicații despre scrierea README-ului și **baremul folosit la corectarea temei**. Vă incurajăm să treceți prin el cel puțin odată.
- Vă recomandăm să folosiți **README.md**, deoarece vă puteți formața conținutul mai bine.

Depunctările pentru temă sunt acoperite în barem, totuși pot apărea depunctări **care nu se află în imaginile furnizate**.

**Bonusuri:** La evaluare, putem oferi bonusuri pentru design foarte bun, cod bine documentat dar și pentru diverse elemente suplimentare.

- **Temele vor fi testate împotriva plagiaturii**. Orice tentativă de copiere va duce la **anularea punctajului** de pe parcursul semestrului și **repetarea materiei** atât pentru sursă(e) cât și pentru destinație(ii), fără excepție.
- Puteți folosi LLM-uri (ChatGPT, Gemini etc.) doar pentru clarificări sau fragmente mici de cod, însă **menționați în README** unde și cum au fost folosite pentru a evita orice scenariu de copiat.

## Git

Folosirea git pentru versionare va fi verificată din folderul `.git` pe care trebuie să îl includeți în arhiva temei. Punctajul se va acorda dacă ați făcut **minim 3 commit-uri relevante și cu mesaj sugestiv**. **NU** este permis să aveți repository-urile de git publice până la deadline-ul hard.

Pentru a primi punctajul pentru **Git**, după ce ați terminat tema și ați făcut toate commit-urile, executați comanda `git log > git_log.txt` în rădăcina proiectului și adăugați fisierul în arhiva trimisă.

Pentru folosirea tool-ului **Git** vă punem la dispoziție un tutorial actualizat și amplu și aveți de asemenea și un tutorial despre comenzi pe care puteți să le dați din IntelliJ la acest.

- După ce clonați repo-ul de pe GitHub, vă rugăm să vă faceți un repository propriu **privat** cu conținutul temei. Dacă nu folosiți un repository propriu **nu o să puteți să urcați schimbările din Git în GitHub**, deoarece vă aflați în rădăcina repository-ului echipei de POO și **aveți riscul să vă expuneți soluția**.
- Vă rugăm să vă generați un **.gitignore** cu template-ul **Maven** dacă nu folosiți gitignore-ul inclus în temă. Acesta vă ajută să definiți fișiere care să nu fie luate în considerare la **diff check** când rulați **git status/git commit** sau alte comenzi de git.

## Checkstyle

Unul din obiectivele temei este învățarea respectării code-style-ului limbajului pe care îl folosiți. Aceste convenții (de exemplu cum numiți fișierele, clasele, variabilele, cum indentați) sunt verificate pentru temă de către tool-ul checkstyle [<https://checkstyle.sourceforge.io/>] (care se află în pom.xml ca dependentă).

Pe pagina de [Recomandări cod](#) găsiți câteva exemple de coding style.

Dacă numărul de erori depistate de checkstyle depășește 30, atunci punctele pentru coding-style nu vor fi acordate. Dacă punctajul este negativ, acesta se **trunchiază la 0**.

Exemple:

- punctaj\_total = 100 și nr\_erori = 200 ⇒ nota\_finala = 90
- punctaj\_total = 100 și nr\_erori = 29 ⇒ nota\_finala = 100
- punctaj\_total = 80 și nr\_erori = 30 ⇒ nota\_finala = 80
- punctaj\_total = 80 și nr\_erori = 31 ⇒ nota\_finala = 70

## Teste

1. test01\_initialize\_entities.json
2. test02\_initialize\_entities\_errors.json
3. test03\_move\_robot.json
4. test04\_move\_robot\_errors.json
5. test05\_env\_condition.json
6. test06\_update\_battery.json
7. test07\_update\_battery\_errors.json
8. test08\_change\_weather.json
9. test09\_scan\_plant.json
10. test10\_scan\_water.json
11. test11\_scan\_animal.json
12. test12\_scan\_object\_errors.json
13. test13\_learn\_fact.json
14. test14\_improve\_environment.json
15. test15\_improve\_environment\_errors.json
16. test16\_medium.json
17. test17\_multiple\_simulations.json
18. test18\_multiple\_simulations\_errors.json
19. test19\_complex.json
20. test20\_complex\_errors.json
21. test21\_complex\_combined.json

Unele teste nu sunt încă disponibile în checker. Acestea vor fi publicate în zilele următoare.

## Upload temă

---

Arhiva o veți urca pe Code Devmind, unde sunt și informații despre structura ei.

Checker-ul din cloud **nu a fost publicat încă!** Vă vom comunica pe Teams când acesta devine disponibil.

## FAQ

---

**Q:** Pot folosi biblioteca "X"?

**A:** Dacă doriți să folosiți o anumită bibliotecă vă recomandăm să întrebați pe forum, ca apoi să o validăm și să o includem și în pom-ul de pe Code Devmind.

**Q:** Pot folosi GSON în loc de Jackson?

**A:** Sigur, contează doar să aveți output-ul corect și să respectați convențiile de coding style și modularitate.

**Q:** Am descoperit edge case-ul "Y", trebuie să îl tratez?

**A:** Nu. Toate datele necesare pentru soluționarea temei vă sunt date în cerință. Dacă totuși am omis ceva ne puteți contacta pe forum.

**Q:** Pot folosi clase de tip "Enum" pentru constante?

**A:** Da.

**Q:** Ce JDK recomandați?

**A:** 25

Q: Pot să fac în orice ordine testele?

A: Depinde. Testele au fost concepute sa fie cât mai decuplate și să testeze câte o funcționalitate în întregime. Cu toate astea, vă recomandăm să implementați mai întâi testele 01, 02, 03 deoarece reprezintă funcționalitățile de bază pentru rezolvarea următoarelor teste mai complexe.

## Resurse și linkuri utile

---

- Schelet de cod [<https://github.com/oop-pub/schelet-tema1-2025>]
- Forum [<https://curs.upb.ro/2025/mod/forum/view.php?id=43089>]
- [Indicații pentru teme](#)
- [Recomandări coding style & javadoc](#)
- Tutorial Git [<https://ocw.cs.pub.ro/courses/poo-ca-cd/resurse-utile/tutorial-git>]

poo-ca-cd/teme/2025/8e211bfe-f6eb-467c-9e54-f8e6df4c1535/tema-1.txt · Last modified: 2025/11/28 21:05 by iulia.popescu2012