UNIVERSIDADE FEDERAL DO CEARÁ – UFC SOBRAL
TÉCNICAS DE PROGRAMAÇÃO – PROF. FISCHER

Prova 1: Sistema JavaLar

A Prova 1 consiste na **implementação** das funcionalidades abaixo descritas e **explicação** do código implementado. O trabalho a ser desenvolvido é um sistema que simula o movimento de rotação e translação dos planetas-linguagens que orbitam no sistema JavaLar.
O sistema deverá ser feito apenas com terminal, ou seja, sem interface gráfica.

Observação importante: **o trabalho não é sobre o sistema Solar**[1].

A estrela do sistema JavaLar e a linguagem Java, e os planetas que integram a esse sistema são, nessa ordem: Python, JavaScript, Ruby on Rails, PHP, C#, C++ e C.

O espaço onde os planetas se encontram está organizado é um plano cartesiano de coordenadas X e Y. Onde X e Y são valores inteiros positivos. A estrela do sistema é o Java, localizado no centro do plano cartesiano. Os planetas orbitam Java em trajetórias em formato de um quadrado, conforme descrito na Figura 1.

[1] https://www.youtube.com/watch?v=3bE0NZVRwNo

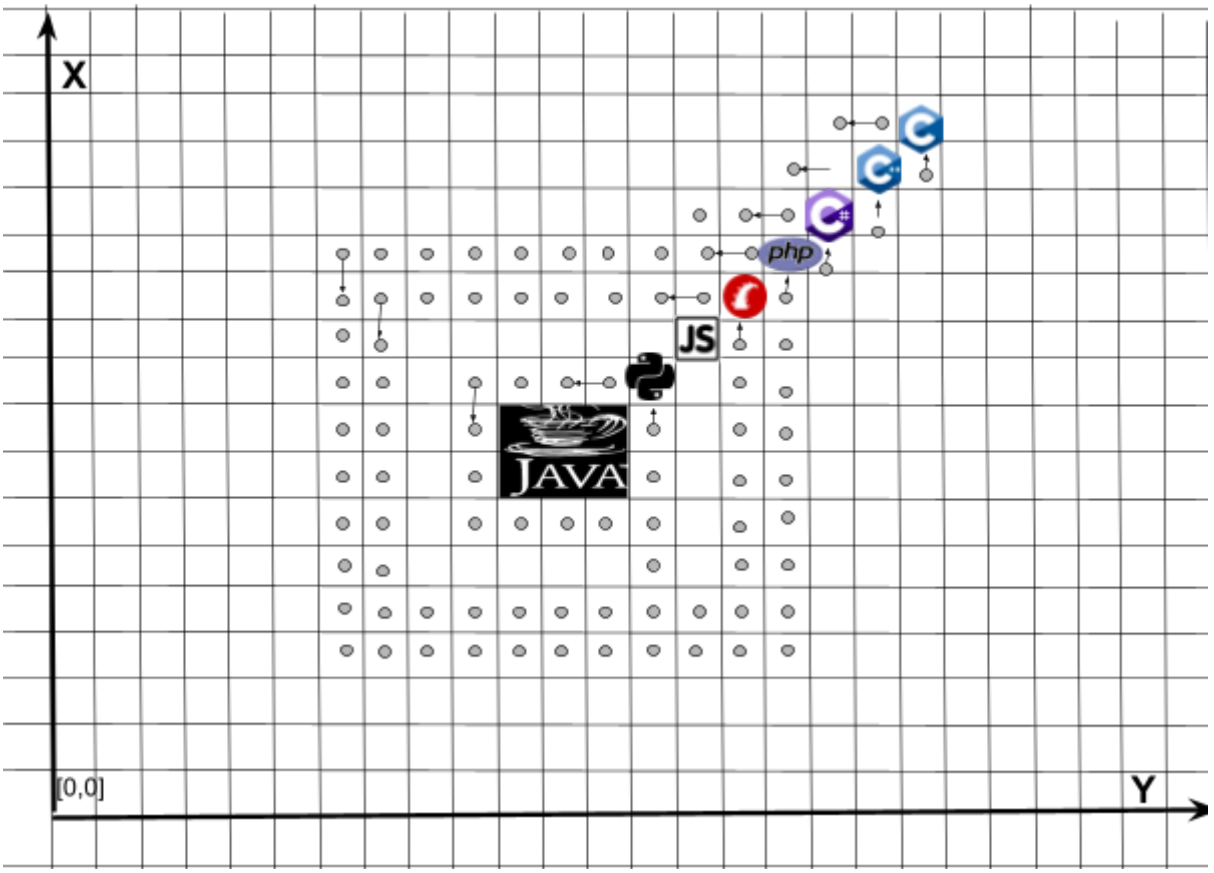

Figura 1: Sistema JavaLar

1. Contexto

Implemente um sistema que possa representar o sistema JavaLar. Inicialmente, a estrela Java e os planetas devem ser posicionados no plano cartesiano. A estrela Java fica no centro do plano cartesiano (fica parada). Os planetas em órbita ficam a partir de Java.
O planeta Python fica X+1 em relação a Java. O JavaScript fica X+2 em relação a Java. O planeta Ruby on Rails fica X+3 em relação a Java. Assim, sucessivamente até chegar o planeta C que fica X+7 em relação a Java. A Figura 2 apresenta as posições iniciais dos planetas.

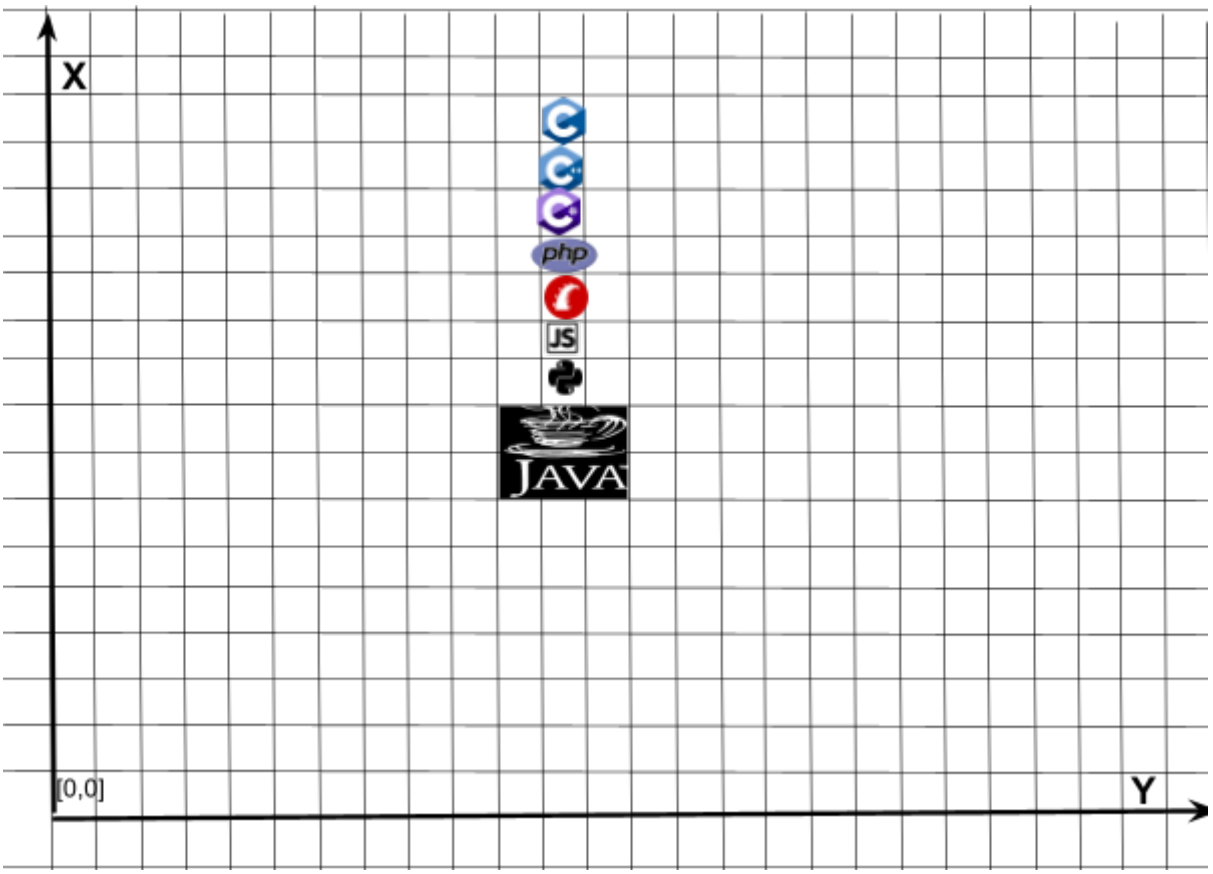

Figura 2: Posição inicial dos planetas

O sistema armazena as posições dos planetas em determinados instantes, como se fosse fotografias tiradas por um telescópio, o usuário que informa o instante, com a quantidade de tempo, por meio de uma opção do menu de usuários.
Os planetas tem velocidade de translação diferente, segundo a descrição abaixo, em cada instante de observação.

- Python: a cada instante de observação se desloca em 4 unidades.
- JavaScript: a cada instante de observação se desloca em 3 unidades.
- Ruby on Rails: 2 unidades
- PHP: 2 unidades
- C#: 1 unidade
- C++: 2 unidades
- C: 10 unidades

O sistema deverá receber do usuário a quantidade de tempo que ele deseja para cada instante. O instante é como uma rodada no sistema.
O deslocamento dos planetas deve ser feito conforme a regra de translação de cada. Assim, o sistema deverá fazer o cálculo considerando a quantidade de tempo informado pelo usuário (a unidade do tempo é irrelevante para o problema). Se for

informado o tempo igual a um (1), então, os planetas se movem conforme suas regras de movimentação, caso o tempo seja 2 os planetas se moveram sua regra mas multiplicado por dois. Por exemplo, PHP move 2 unidades, se o tempo informado for 2 então ele vai se mover 4 unidades, conforme demonstrado pela Figura 3. O movimento de translação é sempre em sentido antihorário de cima para baixo, o movimento é um formato de um quadrado em torno de Java.

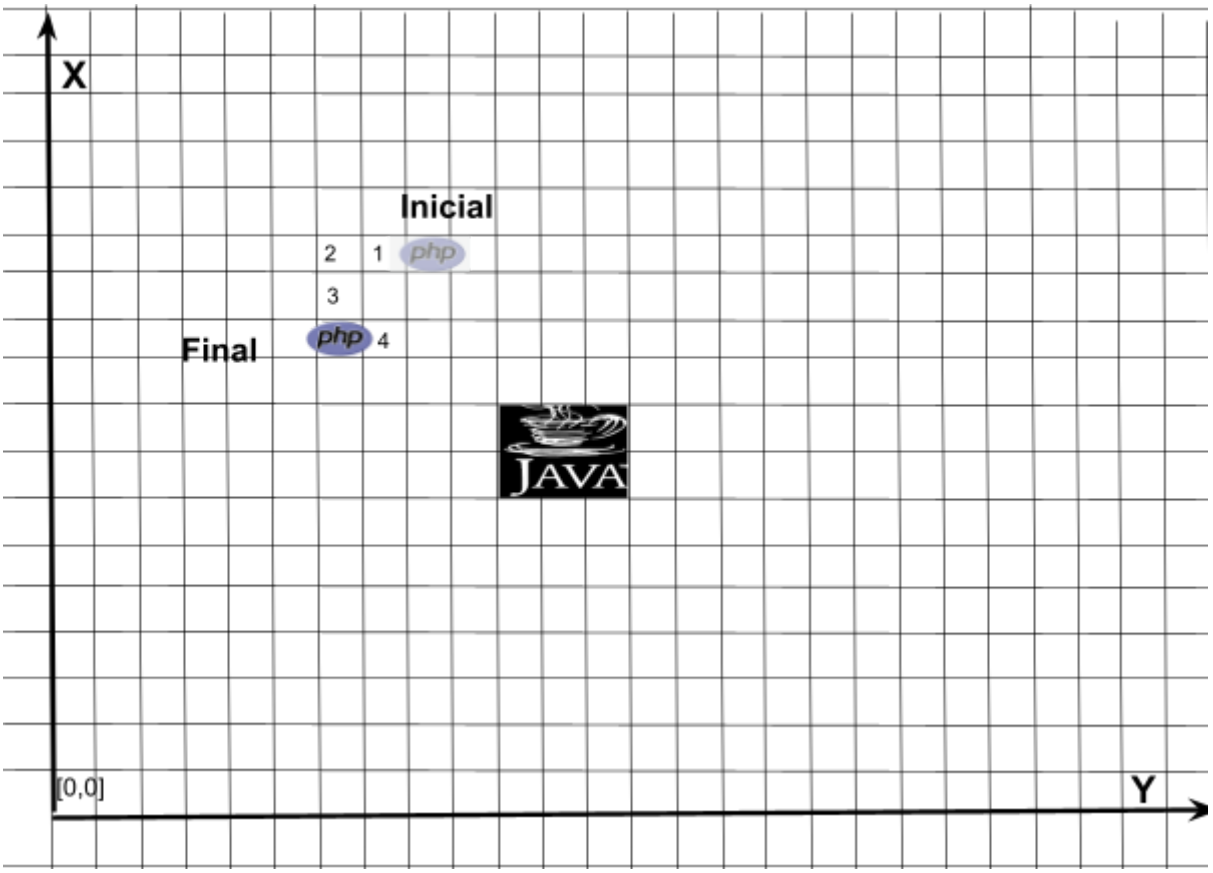

Figura 3: Movimento de translação dos planetas

A cada tempo informado os planetas devem ser posicionados em suas devidas posições no plano, essas posições X e Y devem ser armazenadas pelo sistema a cada instante.

Atenção os planetas nunca se colidem.

Os planetas também fazem o movimento de rotação conforme a listagem abaixo:

- Python: a cada instante de observação se desloca corresponde 24 horas (1 dia).
- JavaScript: a cada instante de observação corresponde 10 horas.
- Ruby on Rails: a cada instante de observação corresponde 48 horas (2 dias).
- PHP: a cada instante de observação corresponde 60 horas.

- C#: a cada instante de observação corresponde 4 horas.
- C++: a cada instante de observação corresponde 0,5 horas.
- C: a cada instante de observação corresponde 0,1 horas em JavaScript.

Como nem tudo é perfeito, existem bugs que podem aparecer no sistema JavaLar e atrapalhar a velocidade normal de translação de cada planeta. Assim, se um bug colidir com uma planeta faz com que o planeta perca uma unidade de velocidade. Porém, se aparecer um desenvolver o planeta acelera em uma unidade.

Planetas que ficam com a velocidade **igual a zero** explodem no sistema JavaLar, e não mais ficam em órbita.

Atenção, Java não morre! Fica a dica.

Tanto os bugs como desenvolvedores são inseridos no plano por meio do sorteio, mas o usuário informa apenas a quantidade de bugs e de desenvolvedores.

2. Funcionalidades do sistema

1) O sistema deverá ter um menu onde o usuário pode indicar um novo instante, a quantidade de tempo do instante, além de escolher o número de bugs e número de desenvolvedores, para cada rodada.
2) O sistema deverá calcular para cada planeta a velocidade de translação que o planeta terá após o tempo do instante informado.
3) O sistema deverá informar quantos dias (em horas) se passaram em cada planeta e **anos JavaLar** para cada tempo informado.
4) O sistema deverá mostrar após cada instante o número de desenvolvedores e número de bugs que foram inseridos em todas as rodadas, bem como sua posição X, Y. O bug e o desenvolvedor inseridos não saem da posição até que colida com algum planeta. Caso um planeta se colida com um bug ou desenvolvedor, o bug ou desenvolvedor deverá ser removido do plano. Então, como cada rodada podem ser colocados novos bugs e desenvolvedores ou eles podem colidir com os planetas o número deles pode crescer ou diminuir.
5) O sistema deverá mostrar após a computação de cada instante as seguintes informações:
    a) Número de planetas no norte (acima de Java) e número de planetas ao sul (abaixo de Java)
    b) Ocorrência de alinhamento dos planetas, quando os planetas ficam em linha. A Figura 4 apresenta as possibilidades de alinhamento.

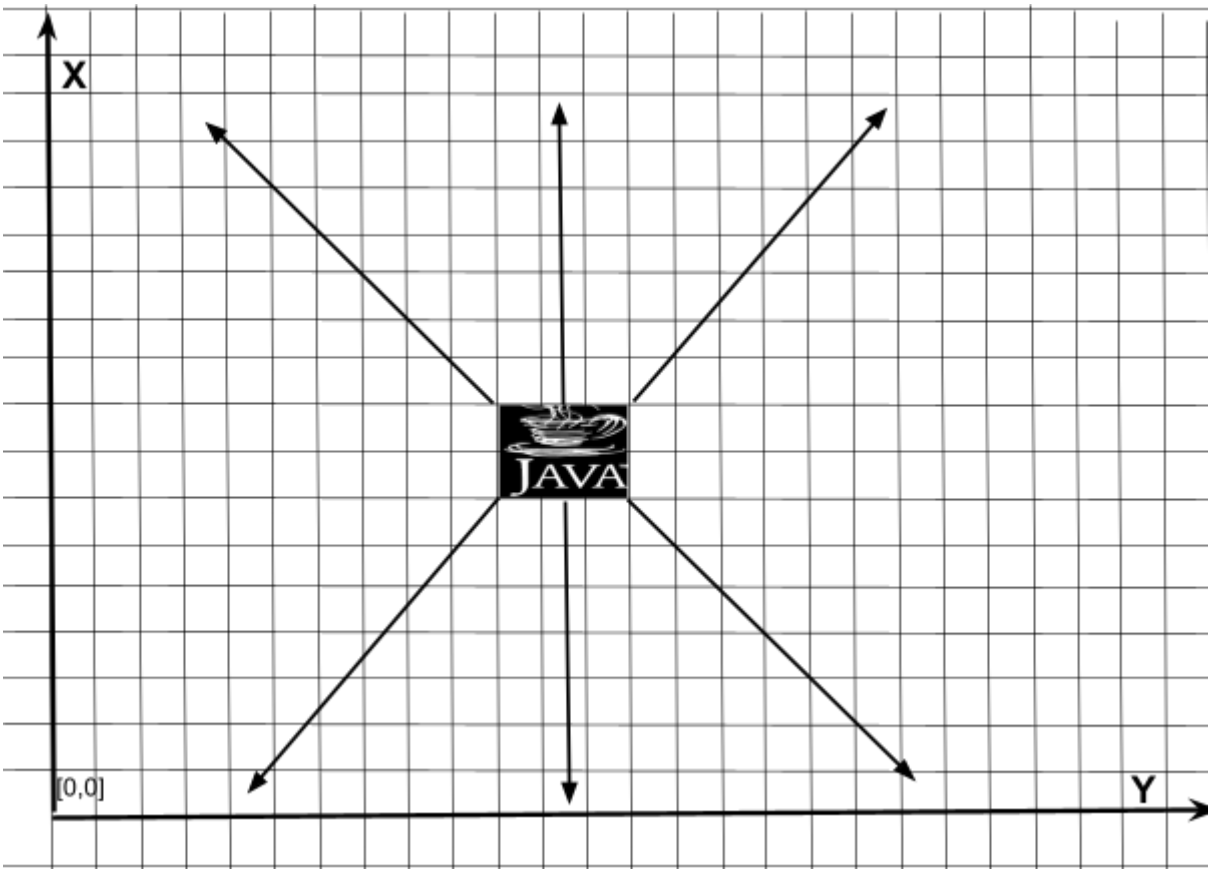

Figura 4: possibilidade de alinhamento

c) Distância (área) entre os planetas (todos para todos), que é o cálculo de distância entre cada planeta, por exemplo a distância de Python para PHP, em um dado instante foi de 20. Dado pela fórmula: largura * altura. A Figura 5 apresenta um exemplo do cálculo de distância.

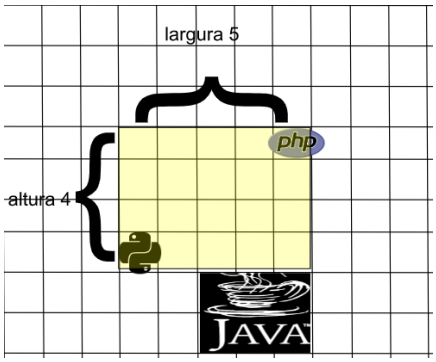

Figura 5: cálculo de distância entre os planetas

d) Distância euclidiana[2] entre cada planeta do sistemas JavaLar

6) O sistema deverá mostrar a velocidade de translação de cada planeta ao final de cada instante.
7) Quando o usuário resolver sair do sistema um relatório deverá ser apresentado, composto das seguintes informações:

[2] https://pt.wikipedia.org/wiki/Dist%C3%A2ncia_euclidiana

a) Número de bugs e desenvolvedores que cada planeta colidiu ao longo da execução do sistema
b) Nome velocidade de rotação e velocidade média de translação de cada planeta
c) Se teve algum planeta que explodiu ao longo da execução do sistema
d) Quantos instantes foram solicitados pelo usuário
e) Quantos dias (em horas) se passaram em cada planeta e **anos JavaLar**.
f) O sistema deverá apresentar uma pequeno resumo sobre cada planeta-linguagem pertencentes ao JavaLar, então faça um pequeno texto sobre cada linguagem e coloque no sistema para que o usuário do sistema tenha acesso.

8) **Questão bônus! O sistema deverá calcular a área de cobertura dos planetas no plano. O cálculo da área deve ser feito contando todos os pares ordenados X e Y que estão estão cercados pelos planetas mais extremos. Como demonstrado pela Figura 6**

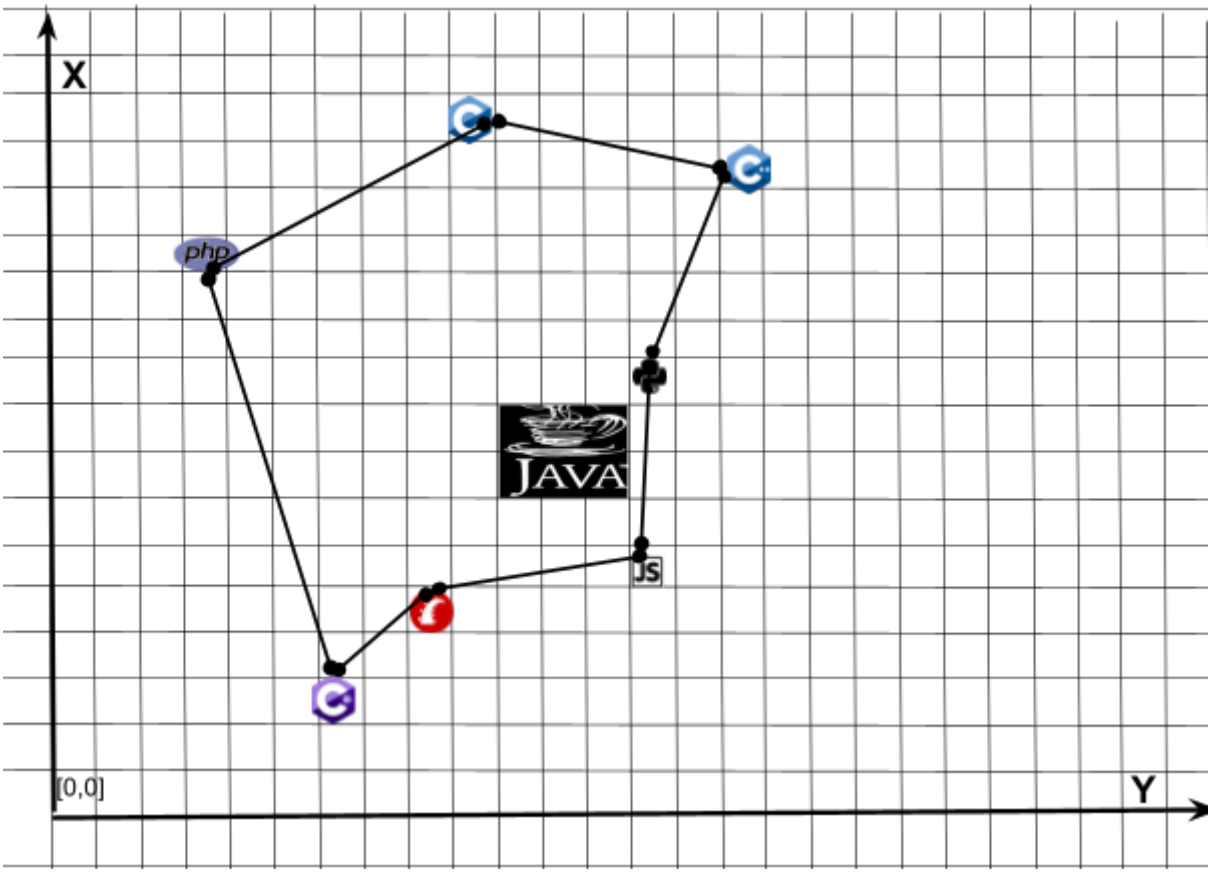

Figura 6: área de cobertura

3. Requisitos para a entrega da Prova 1
   - Data: **08/10/2023**

- Pontuação: **30 pontos**
- Trabalho individual
- Poste seu código no GitHub
- Faça um vídeo de explicação e posto no YouTube como não listado
- Coloque no classroom o link do seu projeto no GitHub e link do vídeo de explicação do seu trabalho
- Você deverá gravar um vídeo entre 13 a 15 minutos explicando o seu trabalho. As apresentações fora desse intervalo (13 a 15 minutos) não serão pontuadas
- Exemplos de ferramentas para gravação: ActivePresenter, câmera do celular e etc…
- No início do vídeo você deverá se apresentar (nome, matrícula e período em que você está cursando) “filmando seu rosto”
- Demais partes do vídeo podem ser apenas a apresentação da sua tela na qual você estará explicando o seu código

4. Requisitos que o seu trabalho deverá ter:

a) Encapsulamento
b) Construtores
c) Herança
d) Polimorfismo
e) Classe abstrata
f) Coleções em Java: ArrayList
g) Interface OO
h) Sobrecarga de métodos
i) Sobrescrita de métodos

Observação importante:Trabalhos que não serão pontuados:

1. Trabalho que não tenha o vídeo de apresentação
2. Trabalhos duplicados para mais de um aluno
3. Trabalhos que não sigam as instruções do sistema acima descrito
4. **Trabalhos que não sigam o paradigma da Orientados a Objetos**

Bom trabalho para todos!