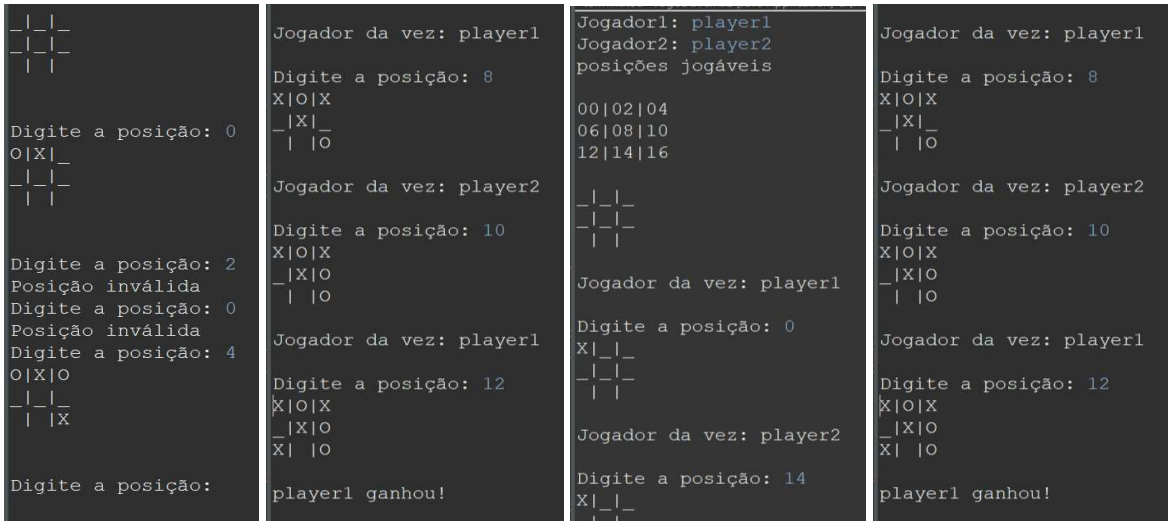
[Ivan] [1/10] [fácil] [array] - Crie a classe Aluno que possuirá o atributo nome, do tipo String. Crie um programa que irá cadastrar 5 alunos em uma matriz 5x4 em que cada linha representa um aluno e cada coluna representa, respectivamente um objeto do tipo Aluno, nota1, nota2 e nota3 do tipo inteiro. As notas devem ser inseridas usando números aleatórios de 0 a 100. Ao final do programa, imprima o nome do aluno, sua média e a sua situação (APROVADO para médias iguais ou maiores que 70, RECUPERAÇÃO se a média for maior ou igual que 50 e menor que 70 e REPROVADO se a nota da média for menor que 50)

[Ivan] [2/10] [médio] [HashMap] - Crie a classe NumeroPorExtenso que terá o atributo mapa, do tipo HashMap (pesquise sobre a classe HashMap) com os parâmetros Integer e String. Essa classe terá também um método privado chamado **popularMapa** que não recebe (por parâmetro) nem retorna nada e irá popular o atributo mapa com um número inteiro e seu equivalente por extenso ex.: (5, “Cinco”) de 1 até 100 (pense em uma forma de reduzir o seu código). Adicione o método intPorExtenso que recebe um inteiro e retorna uma String representando o número por extenso. Por fim, escreva um programa que imprima **aleatoriamente** 20 números por extenso de 1 a 100.

Jogo Da Velha

[Ivan] [3/10] [fácil] [OO] - Crie a classe JogoDaVelha que possui um array de char como atributo que servirá como o tabuleiro, esse atributo deve ser inicializado no construtor e o design pode ser feito conforme o exemplo abaixo. Adicione métodos para validar cada jogada dos dois jogadores e para validar se há um ganhador. Você pode também adicionar outros métodos que achar necessário, bem como fazer a divisão dos métodos listados anteriormente.



[Ivan] [4/10] [fácil] [OO] - Crie um jogo usando a classe anterior, peça o nome dos dois jogadores e informe quem foi o vencedor (se houver) no final do programa. Faça todas as validações para não permitir que o jogador escolha uma posição inválida.

[Ivan] [5/10] [fácil] [Interface] - Crie a classe abstrata Item, que tem os atributos nome e preço, inicialize esses atributos via construtor, não há necessidade de métodos get e set. Crie as interfaces Comida e Bebida, que não terão nenhum método. Crie a classe Suco que

herdará da classe Item e implementa Bebida, não terá métodos adicionais e sempre adiciona “Suco de” ao parâmetro passado via construtor. Crie a classe Pizza, que segue a mesma lógica de Suco, mas implementa Comida e adiciona “Pizza de” ao parâmetro passado via construtor. Por fim, crie um main que terá um array de tamanho 5, e irá cadastrar os pedidos feitos pelo usuário (pizzas e sucos possuem valores fixos). Depois de registrar os pedidos, calcule o total do pedido e imprima o nome de todos os produtos, o preço total com e sem desconto. As bebidas possuem desconto de 5%, e as comidas possuem desconto de 10%.

```
O que você deseja?
1 - Suco
2 - Pizza
Sua escolha: 1
De que sabor? caju
O que você deseja?
1 - Suco
2 - Pizza
Sua escolha: 1
De que sabor? acerola
O que você deseja?
1 - Suco
2 - Pizza
Sua escolha: 2
De que sabor? frango
O que você deseja?
1 - Suco
2 - Pizza
Sua escolha: 2
De que sabor? calabresa

Pedidos:
Suco de caju
Suco de acerola
Pizza de frango
Pizza de calabresa

Total sem desconto: R$ R$ 57,70
Total com desconto: R$ R$ 52,26
```

[Ivan] [6/10] [fácil] [Exceções] - Crie a classe Usuario, que tem os atributos nome e senha. Sobrescreva o equals, que compara os usuários pelo nome, crie os métodos get e set. Em seguida, crie a classe cadastroUsuarios que tem um array de usuários como atributo e recebe o tamanho do array pelo construtor, essa classe terá os métodos addUsuario, sem retorno, que recebe um objeto do tipo Usuario e o adiciona no array, sem permitir usuários repetidos, tem também o método verificarUsuario, sem retorno, que verifica se o usuário passado por parâmetro existe no array, caso já exista, o método lança um UsuarioJaExisteException, com a mensagem “Este nome de usuário já está sendo usado”.

[Ivan] [7/10] [fácil] [Exceções] - No main, crie um programa que peça o tamanho do array e em seguida cadastre os usuários no array, quando for cadastrar a senha, peça para o usuário confirmar a senha e, caso as senhas digitadas forem diferentes, lance uma SenhasDiferentesException com a mensagem “As senhas devem ser iguais!”. Faça com que o cadastro dos usuários continue pedindo os dados de um usuário enquanto os requisitos de cadastro não forem cumpridos. Além disso, faça com que o programa não

perca os dados já inseridos quando alguma exceção for lançada (se o usuário digitar um nome válido e as senhas forem diferentes, peça somente que o usuário digite uma nova senha).

Projeto caça palavras

[Ivan] [8/10] [difícil] [OO] - Crie a classe CacaPalavras, que tem os atributos **letras** (um array de char com as letras do alfabeto) e **tabela** (um array de char). a tabela deve ter tamanho 64 e deve ser preenchida com pontos conforme o exemplo abaixo. Essa classe terá o método sortear, que não tem parâmetro e retorna um char que corresponde a uma letra aleatória do atributo letras. O método privado inverterPalavra, que recebe uma String como parâmetro e a retorna invertida.

[Ivan] [9/10] [difícil] [OO] - A classe CacaPalavras tem também os métodos adicionarLinha e adicionarColuna, que não possuem retorno e receberão uma String, que representa uma palavra, um inteiro, que representa a posição que a palavra será inserida no atributo tabela e um boolean, que determina se a palavra vai ser invertida ou não. Por fim, tem o método gerarTabela que não tem parâmetros e retorna uma String formatada que substitui os pontos restantes da tabela por letras aleatórias e também faz a quebra de linha para formar a tabela. No main, instancie um objeto da classe CacaPalavras, adicione algumas palavras que você escolher em suas posições usando os métodos da classe e imprima a tabela.

```
. . . . . U .
B . . . . . J .
A C E R O L A .
N . . . . . C .
A . . . . . .
N . . . . . .
A I C N A L E M
. . . . . .
```

```
K Z V E R V U P
B X V A K S J F
A C E R O L A F
N H U V U R C E
A R Q A C U V Q
N X Z D H J W I
A I C N A L E M
K Q J U R C R X
```

```
Z I D J D G U Z
B N P E B B J J
A C E R O L A Y
N I J W E T C C
A C L P I W S R
N L O K F S G P
A I C N A L E M
Q R M Y O D M U
```

```java
public static void main(String[] args) {

    CacaPalavras cacaPalavras = new CacaPalavras();
    cacaPalavras.adicionarLinha("acerola", 16, false);
    cacaPalavras.adicionarColuna("banana", 8, false);

    cacaPalavras.adicionarColuna("caju", 6, true);
    cacaPalavras.adicionarLinha("melancia", 48, true);
    cacaPalavras.gerarTabela();

}
```

[Ivan] [10/10] [fácil] [OO] - Qual é a saída para o seguinte código:

a) 132
b) 134
c) 1134
d) 1143
e) erro

```java
public class Classe1 {

    private Classe2 c;

    public Classe1() {
        System.out.print("1");
        c = new Classe2();
    }

    public void doSomething() {
        System.out.print("2");
    }

    private class Classe2 {

        public Classe2() {
            System.out.print("3");
        }

        public void doSomething() {
            System.out.print("4");
        }

    }

    public static void main(String[] args) {
        new Classe1().c.doSomething();
    }
}
```