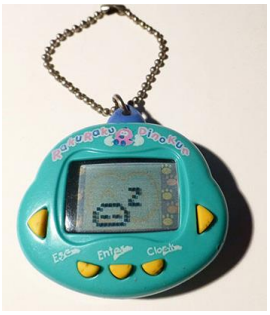
Tamagotchi



O Tamagotchi foi um brinquedo muito usado em meados dos anos 90. O brinquedo consiste em um animalzinho de estimação eletrônico que deve ser cuidado para que não morra.

Assim sendo, você deve criar um programa em Java que simule um Tamagotchi, de acordo com as características e definições descritas no decorrer deste documento.

O que você deve fazer

Você deve fazer um sistema que controle o Tomagoshi do usuário. Assim sendo, você mostrará na tela informações do que está acontecendo com o animalzinho digital e solicitará ao usuário que tome ações de acordo com o que acontece com o animal no decorrer de sua vida.

Um Tamagotchi tem um nome, uma idade e um peso. Ele tem a capacidade de comer, dormir e se exercitar. Neste sentido, ele pode:

- Sentir sono: quando o Tamagotchi sentir sono, duas opções devem ser dadas ao usuário:
  - Dormir
  - Permanecer acordado

  Quando o Tamagotchi avisa que está com sono e permanece acordado 5 vezes seguidas, a próxima vez que ele sentir sono ele dorme automaticamente. Sempre que o Tamagotchi dorme ele aumenta sua idade em 1 dia. Quando o Tamagotchi chega a 15 dias, ele morre.

- Sentir fome: quando o Tamagotchi sentir fome, algumas opções devem ser dadas ao usuário. São elas:
  - Comer muito
  - Comer pouco
  - Não comer

  Comer pouco é o normal do Tamagotchi, ele aumenta 1 quilo a cada vez que come pouco. Um Tamagotchi que come muito aumenta 5 quilos, e logo deve dormir. Um Tamagotchi que sente fome e não come emagrece 2 quilos a cada vez. Se o Tamagotchi ultrapassar os 20 quilos ele explode. Se o Tamagotchi chegar a zero quilos, ele fica desnutrido e morre.

- Ficar entediado: quando o Tamagotchi fica entediado, ele deve se exercitar. Neste caso, devem ser apresentadas duas opções de treino para o Tamagotchi:
  - Correr 10 minutos
  - Caminhar 10 minutos

  Depois de correr por 10 minutos, o Tamagotchi automaticamente emagrece 4 quilos e come muito. Depois de caminhar, o Tamagotchi sempre emagrece 1 quilo e fica com fome.

Como escolher os desejos do Tamagotchi?

Os desejos do Tamagotchi (sentir sono, sentir fome e ficar entediado) acontecem aleatoriamente. Use o método Math.random() para realizar um sorteio entre os valores desejados. Pesquise o funcionamento do método Math.random() e utilize no seu trabalho da forma que achar pertinente.

Seu programa deve simular a vida toda do Tamagotchi, que começa com 0 dias e 1 quilo. Mostre, a cada ação realizada, o estado do Tamagotchi.

Seu programa deve ter, no mínimo, as seguintes classes: Tamagotchi, Principal e Teclado.

- *classe Tamagotchi*: possui todas as informações (atributos) e métodos necessários para um Tamagotchi, como construtores, métodos de acesso e métodos auxiliares para o desenvolvimento do trabalho;
- *classe Principal*: possui o método main. No método main, deve ser inserido o código que faz a lógica para que o sistema funcione da forma descrita;
- *classe Teclado*: possui métodos para viabilizar a entrada de dados do usuário pelo teclado, para interação com o usuário. Esta classe está pronta e será disponibilizada pelo professor.

Obs.: a classe Teclado será abordada em aula, mas ela (classe Teclado) já está disponível no Moodle, caso você queira utilizá-la antes. Você deve incluí-la no seu projeto e chamar os métodos desejados, conforme exemplificado no projeto ExemploClasseTeclado, disponível no Moodle. Os nomes dos métodos são intuitivos, e exemplos de utilização estão disponíveis no projeto ExemploClasseTeclado.

Fique atento aos seguintes itens:

a) *Comente o seu código*, informando no mínimo o que cada método de suas classes faz. Não esqueça de colocar seu nome em **TODAS** as classes que você for implementar (isto **exclui** a classe Teclado). Para colocar o nome, você deve, simplesmente, colocar um comentário no começo das suas classes com esta informação.
b) *Indente seu código*. Códigos não indentados são mais difíceis de entender, difíceis de encontrar erros, além de receberem desconto na nota final. Pesquise sobre indentação.
c) *Utilize impressões na tela* para informar o que está acontecendo no programa
d) *Faça seu código com clareza*.

Outras informações:

- O trabalho deverá ser implementado utilizando o BlueJ. Para utilizar outra IDE, fale **ANTES** com o professor, para verificar a possibilidade ou não
- Deve ser realizado **individualmente**
- Você deverá entregar uma pasta compactada com <u>todos os arquivos do projeto criado no BlueJ</u>
- Trabalhos copiados de colegas e/ou não realizados pelo aluno receberão zero (todos os envolvidos)

Entrega e apresentação

- A apresentação (para toda a turma) será realizada no dia 05/10/22
- Você deve enviar um ZIP com o projeto BlueJ do trabalho pelo Moodle até as 19h30min do dia 05/10/22 e apresentar o trabalho somente após a entrega no Moodle
- Identifique-se claramente no nome do projeto do BlueJ (nome completo)
- A cada dia de atraso na entrega no Moodle serão descontados 2,0 pontos da nota final do trabalho

- Trabalhos com atraso não serão apresentados ao professor, acarretando na perda de 3,0 pontos pela falta da apresentação.

Avaliação:

- Antes de mais nada:

```
if(código não compila){
     System.out.println("Sem avaliação...");
     nota = zero;
}else
     System.out.println("O que você fez será avaliado!");
```

- Os itens do trabalho serão avaliados da seguinte forma:
    - *Código comentado*: desconto de 1,0 ponto por classe não comentada
    - *Código identado*: desconto de 1,0 ponto por classe não identada
    - *Apresentação para o professor*: apresentações confusas, mal explicadas, erradas e/ou que demonstrem que não foi o aluno que desenvolveu o trabalho, acarretarão em descontos na nota final
    - *Classes conforme enunciado:* 1,5 ponto
    - *Interface com o usuário (mensagens e criatividade)*: 2,5 pontos
    - *Corretude do programa (lógica, ações, mensagens, resultados)*: 6,0 pontos

Dicas:

- Comece pelas classes mais simples. Não tente fazer o mais difícil primeiro.
- Faça um trabalho simples mas correto. Não é necessário criar um programa com centenas de linhas de código. Mais vale um programa que funciona e é simples do que um que é demasiadamente complexo e apresenta falhas.
- Implemente e vá testando suas classes. Não deixe para compilar e verificar se funcionam no final de toda a implementação. Se existirem erros, serão mais difíceis de serem encontrados.
- Pense no programa antes de começar a implementar. Comece a codificar somente quando entender exatamente como ele deve funcionar.
- Não invente regras. Elas estão todas claras no decorrer deste documento.
- Não se assuste com a primeira impressão do trabalho. Você tem plenos conhecimentos e total capacidade de implementá-lo. Basta calma e atenção.
- Não deixe para fazer na última semana.
- Reserve um tempo por dia para pensar no trabalho e implementar alguma parte.
- Qualquer dúvida, mande por e-mail, que serão respondidas na medida do possível.

Boa sorte!