ATIVIDADE 1 - INT. TYPESCRIPT
Foi tranquilo, por enquanto tô lembrando de tudo direitim.

ATIVIDADE 2 - INT. PROCESSING
Também foi de boa, eu já tenho uma familiaridade com o processing e gosto muito, foi bom bom praticar novamente, eu só demoro pq eu fico escrevendo minhas observações entre os códigos pra me ajudar em outros momentos, também demorei um pouqim tentando desenrolar o git mas deu certo.

ATIVIDADE 3 - LOBINHO
Depois dessa int. a processing foi legal fazer essa do lobo, eu nunca tinha usado o p5 com imagens dessa forma. A atividade foi bem importante para firmar na cabeça o movimento de criar uma classe e instanciá-la e a importância do constructor nesse processo.

Para além de todo o processo básico de aprendizagem, eu gostei de entender como funciona a função keypressed e acabei pesquisando e entendendo uma curiosidade que sempre tive que é a de entender como funciona o ascii e unicode. Nunca havia entendido como através dos bits as informações eram recebidas e processadas.

Outro ponto interessante na atividade foi entender como limitar o movimento dos personagens dentro da tela, algo que eu solucionei entendendo a contagem certa de números de linhas e colunas.

ATIVIDADE 4 - MOTOCA

No início eu achei um pouco complicada por já estar me acostumando ao processo usando o p5 e relação visual, mas logo fui me adaptando e entendendo o que fazer.

Algo novo nessa atividade é o uso do to string() para que no terminal a gente possa interagir sem problemas uma possível resposta com os atributos da classe.

Achei interessante o processo de interação entre duas classes que nesse caso foi o de pessoa e moto. Durante o processo eu achei mais interessante colocar a idade como condição na função de subir ao invés de colocar na função dirigir, que aí se a pessoa for maior de 10 anos ela nem sobe na moto.

ATIVIDADE 5 - DIGITAÇÃO

Só entendi agora a importância de fazer a atividade da motoca antes dessa, pois aqui também temos que interagir uma classe com outra, mas tem uma complexidade maior.

Neste caso são 3 classes que eu chamei de bolha, mesa de jogo e jogo. Jogo contém mesa de jogo, e mesa de jogo contém bolhas. Bolha tem posição x e y, tem letras, diâmetro, vitalidade

e uma velocidade. Mesa de jogo tem bolhas, tempo de cada bolha percorrer o quadro, tem um contador e também atributos para contar os erros e acertos.

Aqui aprendemos um pouco mais sobre vetores e foi interessante aprender sobre o .push que fica responsável por gerenciar as bolhas, o push. adiciona uma nova bolha ao vetor.

ATIVIDADE 6 - CINEMA

Aqui assim como nos outros exercícios trabalhamos com duas classes, que chamei de cinéfilo e sala de cinema. Cinéfilo tem como atributo nome e cpf e sala de cinema tem como atributo cadeira, que é inicializado no constructor como uma lista. Ainda dentro do constructor a gente coloca um for que através do .push vai sempre criar novas cadeiras de acordo com o número de vagas disponíveis.

Em sala de cinema temos a possibilidade de criar reservas e de fazer cancelamentos.

Em criar reservas se cpf mencionado em novo cadastro for igual a um cpf já cadastrado imprime “cpf já cadastrado” , se cadeira for != de null é porque já está cadastrada com algum cinéfilo e se for = null é porque está vazia e portanto recebe o nome e cpf cadastrado.

Da mesma forma que em reserva, cancelamento recebe um for que percorre a lista e aí se a cadeira mencionada não estiver vazia (diferente de null) e somente se cpf da cadeira tal for igual ao mencionado, então this.cadeiras recebe null, e portanto o espaço na lista que anteriormente estava ocupado fica vazio e a reserva é cancelada.

ATIVIDADE 7 - Pula Pula com menu interativo

Primeiro eu criei a classe criança que tem como atributo nome e idade, depois criei a classe pula pula, que tem crianças na fila e crianças no pula pula. Pra facilitar e já começar com crianças na fila e no pula pula, criei duas listas com várias crianças na fila e no pula pula e chamei de iniciar listas, então no constructor fica só iniciarlistas.

No pula pula temos sair do pula pula onde eu testo através do lenght, onde se o valor de length for menor ou igual a é porque não tem crianças no pula pula, se tiver criança no pula pula, crio a variável criança que vai receber this.criancasnopulapula.find que vai procurar exatamente a criança pelo nome que foi mencionado.

Daí se a variável criança que recebe o nome da criança quer ser buscada não for undefined, através do nome que foi encontrado o .push coloca ela de novo na fila.

No método inserir criancas na fila, eu crio uma nova criança e coloco na fila.

Em entrar no pula pula temos o teste pra saber se há crianças na fila, isso é feito através do lenght de crianças na fila que se for maior que 0 e não for undefined tem crianças na fila, a

variável criança então recebe o nome da primeira criança da fila através do shift e através do .push coloco essa primeira criança da fila no pulapula.

Aí tem a classe criança com os gets e o sets e depois a classe inicializar com o menu.

ATIVIDADE 8 - LAPISEIRA com menu interativo

Nessa atividade trabalhamos com as classes: grafite e lapiseira, depois adicionei o menu interativo parecido com o do tamagoshi.

Seguindo a lógica da questão, lapiseira tem como atributo: calibre, bico, grafite, temgrafitenobico e grafiteemuso. Como métodos tem: Inserirgrafitenotambor, inserir no bico, inserir, remover grafite do bico, remover grafite do tambor, escrever, posso escrever, acabougrafite e gastar grafite. Quando acabou eu percebi que poderia ter feito menos métodos mas separar assim me ajudou a ter mais clareza na construção de cada ação e deu certo.

Em inserirgrafitenotambor primeiro faz um teste pra definir se o calibre é incompativel, se não for incompátivel, a variavel grafite recebe um novo Grafite e adiciona no tambor.

Não é possível adicionar no bico o grafite se ja estiver ocupado, então em inserir no bico eu testo se tem grafite no bico, removo do bico(criei esse método antes lá em baixo), ai removo um novo do tambor (criei tb antes lá em baixo) e através do método inserir eu coloco um novo no bico.

Em escrever, eu testo primeiro se não tem grafite no bico, o que impede de escrever, tb faço um teste se posso escrever através do método posso escrever (que criei antes) onde se o tamanho do grafite em uso for maior que 1, posso escrever se não, informa que não tem tamanho suficiente.

Em escrever tb tem o cálculo do grafite gasto que é o valor de tamanho do grafite em uso * o número de folhas. E embaixo mais um teste onde se acabou grafite eu removo o grafite do bico e o grafite em uso fica vazio.

Em gastar grafite crio a variável tamanho do grafite que recebe o valor do tamanho do grafite em uso menos o grafite gasto, e se esse valor for menor ou igual a 1, não há tamanho suficiente para escrever.

Aí vem a classe grafite, que tem calibe dureza e tamanho com os get e os sets pra eu trabalhar dentro de lapiseira, mas tb tem o gasto por folha que coloquei no padrão proposto pela questão.

ATIVIDADE 9 - TAMAGOTCHI

Inicialmente eu senti uma leve dificuldade em baixar os pacotes, sempre dizia que : “ a execução de scripts foi desabilitada neste sistema”. Quando acontecem esses problemas no vscode eu sempre acho que nunca vou conseguir resolver sozinha, mas dessa vez sempre que algo assim aconteceu passei a buscar mais nos fóruns e sempre encontro alguma solução. Dessa vez encontrei em um fórum que eu podia dar esse comando: Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser, e deu certo :).

No tamagotchi acho que o aprendizado foi mesmo entender o uso do controlador de acesso, que podem ser public, private e protected. Lendo na internet algumas coisas sobre eu vi que o o controlador de acesso no typescript de fato não gera essa restrição e que fica mais como uma sugestão, mas não sei (perguntar pro Davi depois).

Daí é isso acho que o aprendizado aqui pra mim foi mais entender essas dinâmicas entre o get e o set, onde get é o comando que proporciona o acesso ao atributo privado da classe e o set é onde podemos criar alterações ou restrições através das condições para esse atributo.

Mas como um todo a dinâmica foi, colocar tudo em uma única classe que chamei de bichim.
Bichim tem nome, energia,energia max, vivooumorto,higiene,higienemin, alegria e alegriamin.

Ai sai colocando todos os get e set da classe por indicação do prof pra agnt aprender isso.

Pensando que em todos os valores o bicho começa com 100, então:

Em energia eu fiz próximo ao que o david fez, mas adicionando mais uma condição onde ser energia for igual ao valor de energia max - 80, avisa que precisa comer logo ou vai morrer.

Ai fiz um metodo alegria onde, se alegria min for maior que 0 e alegria min menor que 30, informa que o bicho ta triste.
Outro metodo higiene onde se higiene for maior que 0 e menor que 30 informa que o bicho tá sujo. Além disso tem o método banho, que quando é chamado aumenta higiene, energia e alegria. E o comer, que quando come, aumenta energia e alegria
Tem também brincar, que quando brinca diminui energia, aumenta alegria e diminui higiene.

Aí depois disso tem o menu de inicialização que fiz bem próximo ao do prof mesmo, e acho que vou tentar fazer depois no lapiseira e no pula pula.

ATIVIDADE 10 - Contato com menu interativo - qual problema aqui ? (telefones em lote)

Nessa atividade começamos com uma classe contato, que tem nome e fones
Tem como método adicionartelefone, removertelefone e validade telefone, além dos gets e sets. Antes de implementar o adicionar foi importante fazer o valida telefone, que usando o RegExp eu achei isso na internet como uma solução que nesse tipo de caso faz sentindo e consegui colocar como uma forma de regular o número de telefone para que fique sempre no formato 88 - 88884444 ou 88 9 8884 5245.

No final tem o validartelefone.test que retorna um valor boolean pra indicar se realmente o padrão que eu defini corresponde com o valor colocado.

Ai pra adicionar e remover foi mais simples, só usando o .push e pra remover o .splice

Aí em telefone eu tenho o label e o número, os gets e os sets.

E também fiz o menu interativo.

ATIVIDADE 11 - Busca (problema aqui é a duvida se a pesquisa ta do jeito que a questao quer)

Nesta atividade eu tive mais dificuldade em desenvolver a parte da busca que além do nome, busca também o id e o número do telefone.

Essa atividade dá uma continuidade com a do contato. Na lógica onde ém telefones contém label e número, contato contém telefone, e agenda contém contato.

Então aproveitei a mesma estrutura da questão do contato e em cima desenvolvi a agenda.

A agenda tem como atributo contatos, que é uma lista de contatos, onde o foreach percorre a lista de contatos e se o nome que é definido como chave, for igual ao nome mencionado, retorna contato.

Outro método de agenda é add contato, onde crio a variável nome que recebe um getnome de contato, uma variável fone, que recebe os telefones de contato,e a variável existente que recebe o teste que criei como método lá em baixo, chamado contatoExiste, dai se a variável existente não for vazio, o contato é adicionado. (é se for vazio ou se não for vazio ?)

O método contatoexiste que chamo dentro de add contato, é basicamente um for que percorre a lista de contatos, e se o contato pego através da chave nome, for igual a nome, o contato existe.

O método remover contato é a mesma estrutura do for percorrendo a lista e se nome foi igual ao nome mencionado pra ser removido, ele é removido através do splice.

Atividade 12 - Busca e cinema com mapas

Vou deixar pra fazer depois de herança e hospital por que são questões que valem mais.

Atividade 13 - Herança Criativa

Nesta atividade eu fiquei meio confusa com o lance de ter que dividir em abas diferentes o restante do código, eu fiz isso por indicação do leonardo que me ajudou e tb de vídeos que vi na internet, além disso eu tb fiquei meio emperrada pensando como pegar a hora atual do usuário e usar isso dentro do código, mas descobri que existe o objeto Date. que faz isso. Mas fora isso eu achei bem tranquilo.

Dai eu fiz a classe abstrata plantas que tem: nome popular, nome científico e cor como atributos padrão. E as classes filhas, tem como atributos, altura atual, altura média e máxima, dia aguado e tempo de aguar e como métodos tem o "pode podar", que é usado quando a altura máxima for atingida, o "podar" que é quando caso a altura máxima seja atingida a altura atual recebe a altura média.

Tem também o "pode aguar" onde eu crio uma variável que recebe o valor da data atual menos um número que representa o último dia em que a planta foi aguada, que no constructor já recebe a data atual. E por fim o aguar que informa se está regando ou se não pode aguar.

Atividade 15 - Hospital

Eu fiquei levemente confundido o conceito de interface com o de classe abstrata, mas foi ótimo entender as diferenças nessa confusão.

DO que eu entendi, a interface nem é um classe, tipo nela só vais os métodos mesmo, não tem construtor nem nada, é só os atributos e métodos que as classes que vão partir dessa interface vão obrigatoriamente ter que ter e aí começou a fazer mais sentido pq na questão médicos tem que ter um paciente, um nome e uma especialidade e paciente sempre tem que ter um médico, um nome e um diagnóstico.

Ai pronto fluiu mais ai em médico eu tenho os métodos adicionar e remover paciente, e em paciente, adicionar e remover médicos.

terminar relatório
organizar pastas
colocar no github
revisar conceitos teóricos