Universidade do Estado de Santa Catarina
Ciência da Computação - Programação Orientada a Objetos
Prova I – 2018 / 2

Nome: _____ **Nota:** _____

Implemente o mini-projeto apresentado a seguir utilizando o paradigma de orientação a objetos da melhor forma possível. **Documente** os métodos implementados (com exceção dos *getters* e *setters*).

Super Trunfo é um jogo de cartas que se popularizou bastante no Brasil. Neste jogo, precisa-se de um baralho de cartas Super Trunfo e pelo menos dois jogadores. As *Trunfo Cards* representam alguma coisa qualquer do mundo real, sendo que cada baralho utiliza uma temática específica, como por exemplo: tanques, aviões, motos, carros e personagens. Cada carta possui seu nome e código único (identificador da carta) e uma lista de valores que são utilizados efetivamente para o jogo. Para esta prova você deverá modelar um jogo de Super Trunfo Grand Turismo (cartas referentes a carros).

No jogo de Super Trunfo Grand Turismo (STGT) existe uma coleção de 40 *Trunfo Cards* sobre carros que, além do nome e do código previamente mencionados, possuem a seguinte lista de valores: classe do carro, potência (hp), tempo de aceleração 0-100km/h e peso (kg). Veja um exemplo de *Trunfo Card* do STGT:

Cod: 102.345 / **Nome**: Jaguar XJ-S / **Classe**: A2
Potência: 287 cv / **Aceleração (0-100km/h)**: 7,1s / **Peso**: 1120 kg

No início de uma partida, as cartas do baralho são igualmente divididas entre dois (ou mais) jogadores que se enfrentam em várias disputas. Cada jogador escolhe um apelido (*nick*) para ser identificado no jogo. Inicialmente um dos jogadores participantes é escolhido (por sorte) como desafiante. Em cada disputa ocorre a seguinte rotina: cada jogador participante saca uma carta e o jogador desafiante escolhe um valor da lista de valores das cartas; os jogadores participantes então comparam o respectivo valor e o jogador que tiver a melhor pontuação (que pode ser a menor ou a maior) ganha todas as cartas que estavam na disputa, adicionando-as no fundo (final) de seu baralho. No STGT utiliza-se a seguinte regra de comparação:

- Para classe, tem-se apenas cinco valores: A1 (melhor), A2, B1, B2, B3 (pior);
- Para potência e peso ganha quem tiver o maior valor;
- Para tempo de aceleração ganha quem tiver o menor valor.

O jogador que ganha uma disputa (rodada), se torna o jogador desafiante que escolherá o valor a ser comparado na próxima disputa, mesmo se ele já era o jogador desafiante na rodada anterior. Se algum jogador ficar sem cartas, este é considerado perdedor e o jogador que tiver todas as cartas é proclamado vencedor.

Você deverá implementar uma versão digital da estrutura e de algumas funcionalidades deste jogo em uma versão modificada e adaptada. O seu sistema deverá representar as *Trunfo Cards* e os jogadores. Para estes, deve-se respeitar os seguintes aspectos e implementar as seguintes operações:

- Salvo exceções, todos os atributos das *Trunfo Cards* são externamente acessíveis, mas não modificáveis (nem mesmo por métodos *setters*);
- O nome de uma *Trunfo Card* deverá conter no mínimo cinco letras, caso contrário, a carta deverá receber como nome a string: "?" (sem aspas);
- A *Trunfo Card* possui obrigatoriamente uma classe que pode assumir os valores: A1, A2, B1, B2 e B3.
- O valor de potência, tempo de aceleração e peso deverão estar dentro de uma faixa aceitável: $1 \leq$ potência ≤ 2000 / $0,1 \leq$ tempo aceleração $\leq 50,0$ / $50 \leq$ peso ≤ 5000;
- Como as cartas são criadas com padrão internacional, o valor de potência das *Trunfo Cards* é passado utilizando a unidade *hp* (*horse power*), mas para a versão do STGT brasileira, deseja-se utilizar este valor em *cv* (cavalo a vapor), sabendo que as unidades possuem a seguinte relação: 1 *cv* = 0,98632 *hp*;
- O código de cada *Trunfo Card* deverá ser gerado unicamente de forma controlada pela própria classe de *Trunfo Card*. Faça com que tal valor seja atribuído de maneira incremental iniciando por 100.001 e aumentando de um em um (i.e. 100.001, 100.002, 100.003, ...);

- Cada *Trunfo Card* deve ter uma *flag* (variável booleana) que indica se a carta possui algum erro referente aos dados que lhe foram atribuídos. Por padrão, a *flag* é falsa e pode ser utilizada. Caso um valor inválido tente ser atribuído ao nome, potência, tempo de aceleração ou peso (qualquer um deles), deverá ser exibida uma mensagem de erro apropriada na saída padrão (console), o valor deverá ser definido como zero (ou "?" para o nome) e a *flag* de erro deverá ser marcada como verdadeira;
- Uma *Trunfo Card* só pode ser criada passando-se os seguintes valores referentes ao carro: nome, classe, potência, tempo de aceleração e peso;
- Implemente um método denominado `toString()` o qual retorna uma String com as informações da carta com um formato semelhante ao exemplo apresentado;
- A classe *Trunfo Card* deve ter uma operação `compara(TrunfoCard,TrunfoCard,TipoDisputa)` no qual duas *Trunfo Cards*, passadas como parâmetro, são comparadas com relação a um de seus valores de acordo com o terceiro parâmetro. Este terceiro parâmetro pode assumir apenas os quatro seguintes valores: disputa por classe, disputa por potência, disputa por aceleração e disputa por peso. A operação deve retornar algum valor identificando quem foi o vencedor;
- Cada jogador possui um apelido (*nick*) e um conjunto (vetor) de *Trunfo Cards* que devem ser privados. A quantidade máxima de *Trunfo Cards* que um jogador pode ter é 40 (defina este valor utilizando uma constante visível em qualquer parte do sistema), isto é, quando um jogador for instanciado, deve-se receber e atribuir o seu apelido e criar um espaço máximo de 40 cartas para ele;
- O apelido de um jogador deve ter pelo menos uma letra e não pode ter como subpalavra o termo "palavrao". Caso isto ocorra, defina-o como "-" (sem aspas) e mostre uma mensagem de erro;
- Para cada jogador não deve ser possível modificar externamente seu apelido e suas *Trunfo Cards* de maneira direta. Deve-se implementar uma operação denominada de `limparDeck()` que desvincula todas as *Trunfo Cards* registradas para o objeto jogador e uma operação denominada de `addCarta(TrunfoCard)` a qual adiciona uma nova carta (passada por parâmetro) ao seu baralho. Estas duas operações devem ser acessíveis em todo o sistema;
- Para os jogadores, implemente uma operação denominada `sacar()` que escolhe, retira e retorna uma carta válida do baralho do jogador de forma aleatória (utilize a classe *Random*) – caso a carta sorteada seja nula (*null*), repita o processo até que uma carta válida seja encontrada;
- Voltando à representação das *Trunfo Cards*, adicione um novo elemento para cada carta: a referência para o seu dono (qual jogador possui aquela carta em um dado momento). Crie os métodos *setters* e *getters* para este novo elemento e, no construtor, defina inicialmente tal valor como nulo. Por fim, atualize as operações `limparDeck()` e `addCarta()` de modo a modificar apropriadamente o dono;
- A classe jogador deve ter uma operação `distribuirCartas(Jogador,Jogador,TrunfoCards[])` que recebe como parâmetro duas instâncias de jogadores e um *array* de *Trunfo Cards*. Este método deve distribuir as cartas do *array* aos jogadores igualmente, com diferença máxima de uma carta (se for número ímpar), atualizando o registro da carta sobre qual jogador está com sua posse;
- A classe jogador deve ter uma operação denominada `disputa(TrunfoCard,TrunfoCard,Item)` na qual é realizada a lógica principal do jogo. Nela, a primeira carta passada como parâmetro é comparada com a segunda carta passada como parâmetro em relação ao item especificado no terceiro e último parâmetro. O vencedor – dono da primeira ou segunda *Trunfo Card* – deverá adicionar no seu baralho ambas as cartas do duelo. Esta operação deve retornar um valor booleano verdadeiro caso o primeiro jogador vença, ou um valor falso caso ele perca.

Gere uma classe principal denominada `Main`. Nela, crie o método principal de execução do programa e inicialmente cria três instâncias de *Trunfo Cards* – duas com dados corretos e uma com dados problemáticos – e mostre seus dados na tela. Crie também duas instâncias de jogador e utilize o método distribuir cartas.

```
if( you.haveStudied() ) printf("--- Boa prova! ---");
else printf("--- Boa sorte! ---");
```