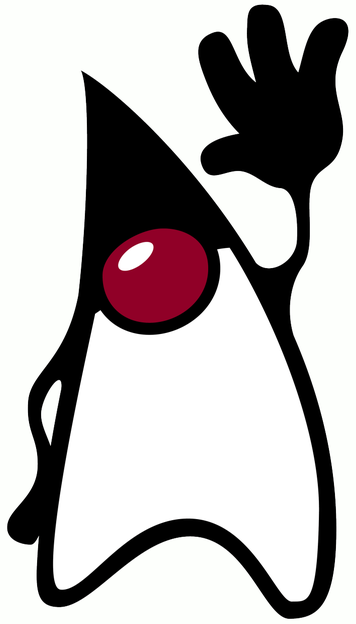
UNIVERSIDADE
FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Classes abstratas

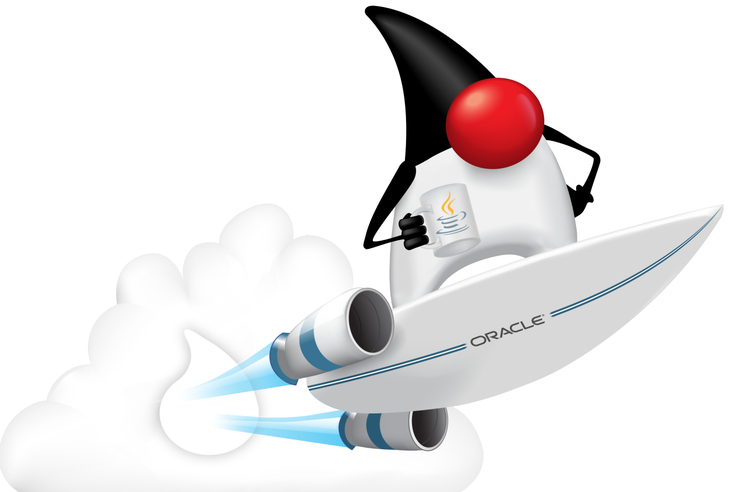
QXD0007 - Programação Orientada a Objetos



Prof. Bruno Góis Mateus (brunomateus@ufc.br)

Conteúdo

- Introdução
- Capturando um Pokemon
- Classe abstrata


ORACLE

Introdução

Introdução

- Contrato
  - Um mecanismo que exige que o desenvolvedor cumpra as especificações
  - Isso inclui seguir nomes de métodos, número de parâmetros e outra coisa mais
- Forçar o cumprimento é vital
  - Um desenvolvedor é sempre capaz de quebrar o contrato
  - Um desenvolvedor iniciante pode resolver reinventar a roda ao invés de utilizar o especificação
- Em Java temos duas opções de definir esses contratos: Classes Abstratas e Interfaces

Capturando um pokemon

Capturando um pokemon

$$X = max(\frac{(3 * HP_{max} - 2 * HP_{atual}) * TAXA_{modificada}}{3 * HP_{max}}, 1) + BONUS_{estado}$$

[1, 255]

Pokédex data

National №	**147**
Type	DRAGON
Species	Dragon Pokémon
Height	1.8 m (5′11″)
Weight	3.3 kg (7.3 lbs)

Training

EV yield	1 Attack
Catch rate	45 (5.9% with PokéBall, full HP)
Base Friendship	35 (lower than normal)
Base Exp.	60
Growth Rate	Slow

Base stats

HP	41		192	286
Attack	64		119	249
Defense	45		85	207
Sp. Atk	50		94	218
Sp. Def	50		94	218
Speed	50		94	218
Total	**300**		Min	Max

Mais informações: https://www.dragonflycave.com/mechanics/gen-ii-capturing e https://bulbapedia.bulbagarden.net/wiki/Poké_Ball

Capturando um pokemon

Training

EV yield	1 Attack
Catch rate	45 (5.9% with PokéBall, full HP)
Base Friendship	35 (lower than normal)
Base Exp.	60
Growth Rate	Slow

$$TAXA_{modificada} = f(x)$$

 $f(x) = x$

 $f(x) = 1.5 * x$

 $f(x) = 2 * x$

 $f(x) = \begin{cases} 4 * x & \text{Se speed do Pokemon} > 100 \\ x & \text{caso contrário} \end{cases}$

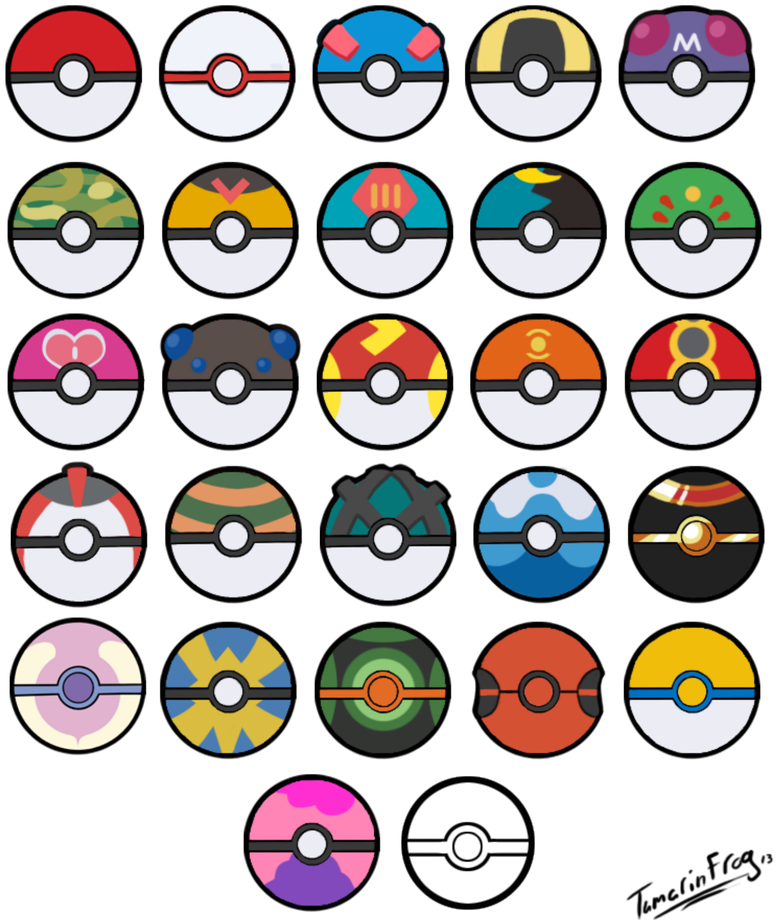
 $f(x) = \begin{cases} 8 * x & \text{Se o Pokemon é do sexo oposto e da mesma espécie} \\ x & \text{caso contrário} \end{cases}$

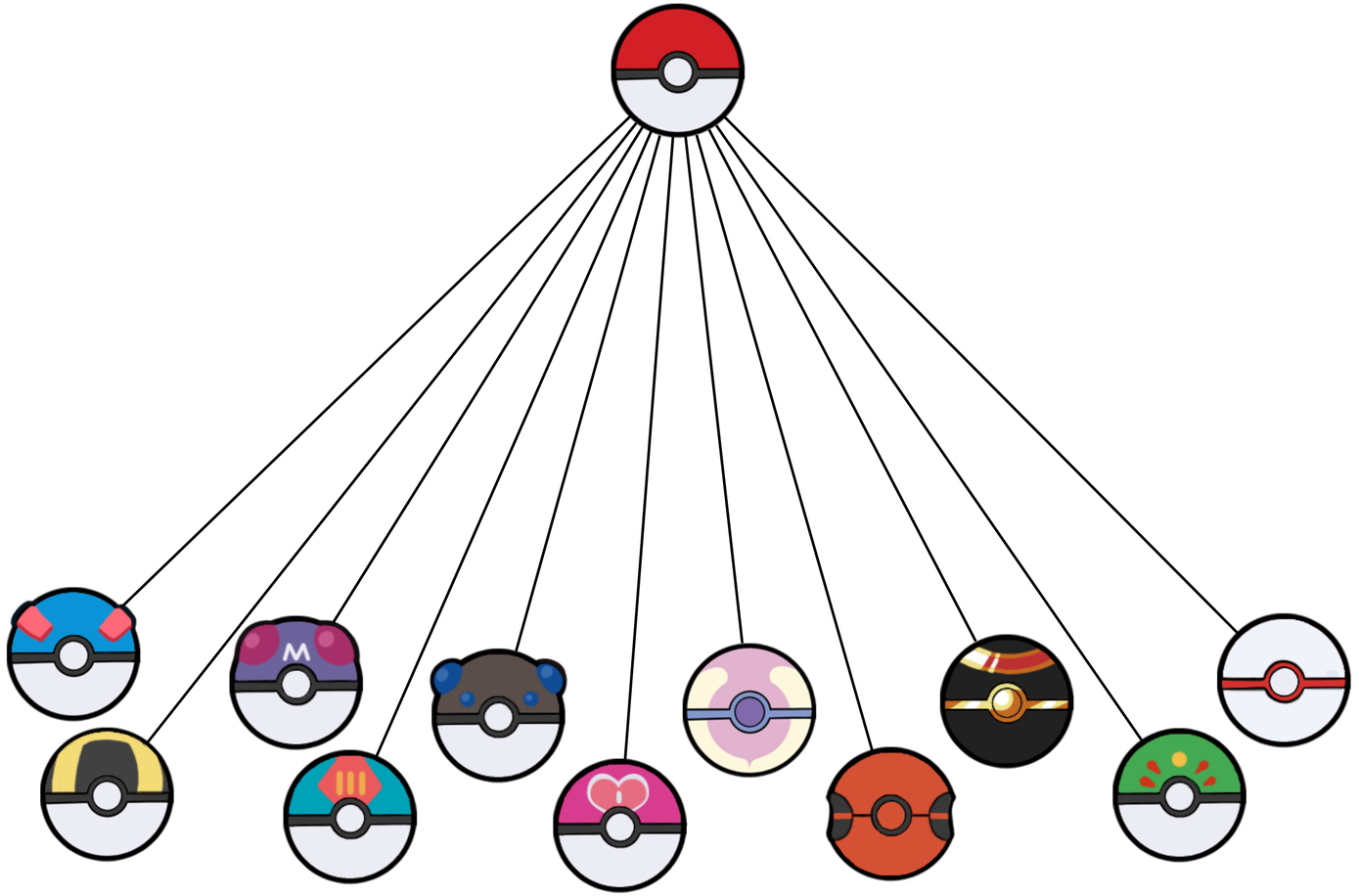
 $f(x) = \begin{cases} -20 & \text{Se peso} \leq 99 \\ x & \text{Se peso} \geq 100 \text{ e} \leq 199.9 \\ +20 & \text{Se peso} \geq 200 \text{ e} \leq 299.9 \\ +30 & \text{Se peso} \geq 300 \end{cases}$

 $f(x) = \begin{cases} 5 * x & \text{Se o pokemon foi pescado} \\ x & \text{caso contrário} \end{cases}$

Capturando um pokemon

Reuso sem subtipos





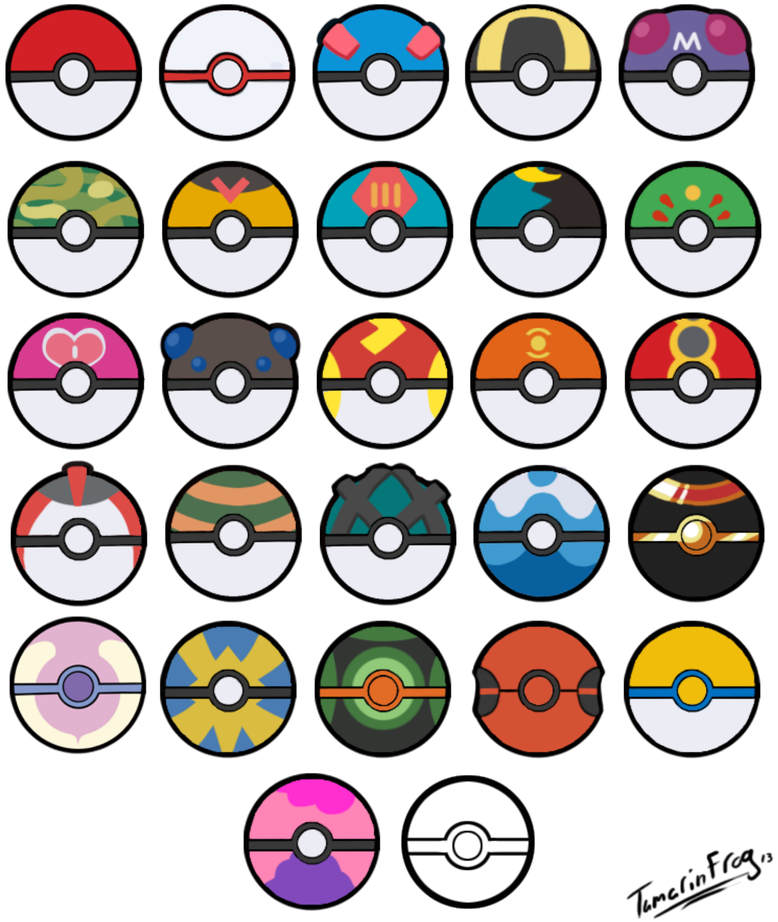
Capturando um pokemon

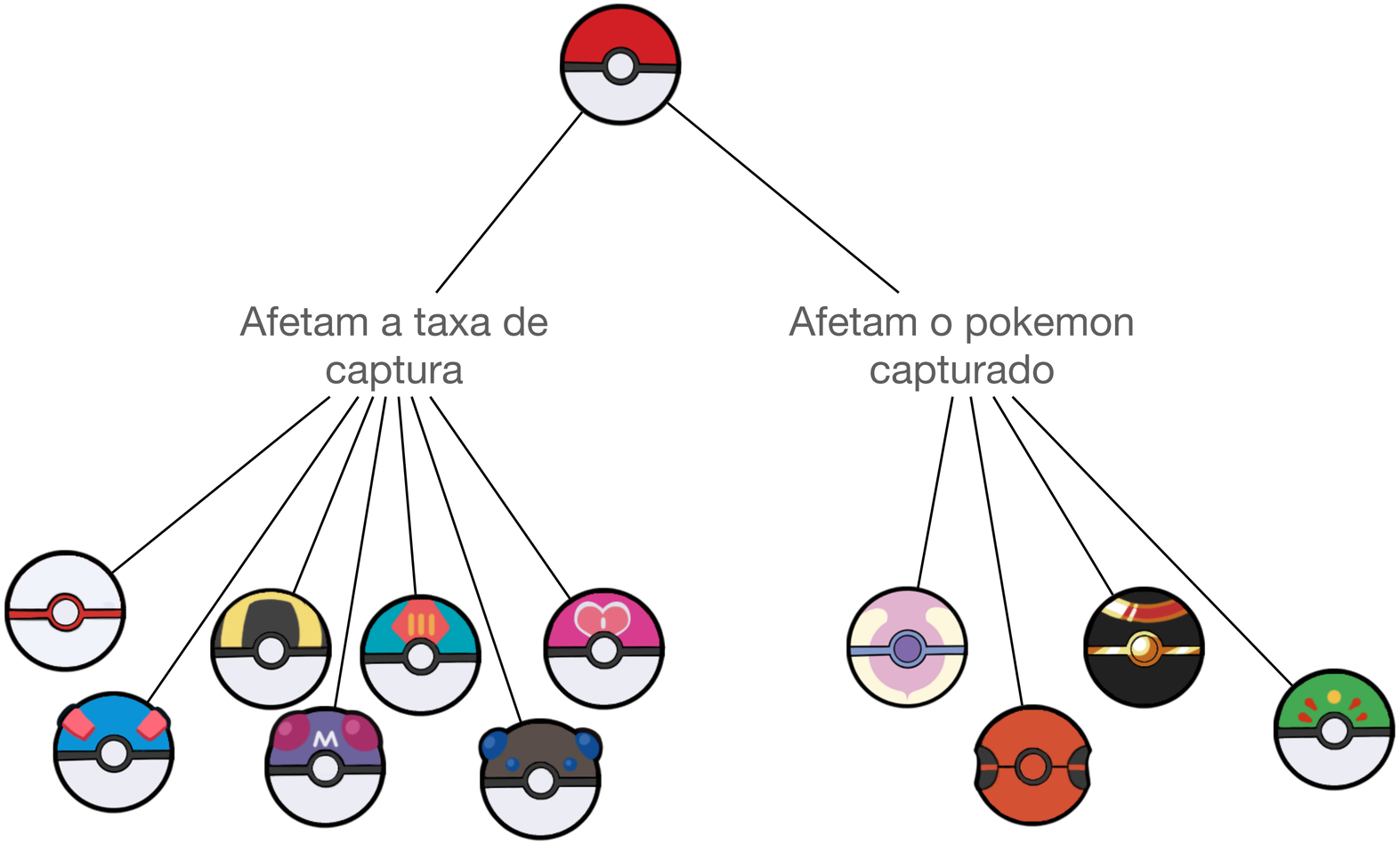
Subclasses, comportamento e evolução

- Objetos da subclasse comportam-se como os objetos da superclasse
- Redefinições de métodos devem preservar o comportamento (semântica) do método original
- Deveria ser possível raciocinar sobre o código usando-se apenas a definição dos tipos das variáveis envolvidas
- O comportamento do código deveria ser independente do tipo do objeto associado a uma dada variável em tempo de execução
- Grande impacto sobre manutenção/evolução de software…

Capturando um pokemon

Reuso com subtipos





Classes abstratas

Classes abstratas

- Possibilita herança de código preservando comportamento (semântica)
- Métodos abstratos
  - Geralmente existe pelo menos um
  - São implementados nas subclasses
- Não pode ser instanciada
  - Mas podem ter construtores, para reuso
  - Atributos qualificados como protected para serem acessados nas subclasses

Classes abstratas

- Herdar código sem quebrar noção de subtipos, preservando o comportamento do supertipo
- Generalizar código, através da abstração de detalhes não relevantes
- Projetar sistemas, definindo as suas arquiteturas e servindo de base para a implementação progressiva dos mesmos

Por hoje é só