IFBA – Instituto Federal de Educação, Ciência e Tecnologia
Análise e Desenvolvimento de Sistemas
INF011 – Padrões de Projeto
Prof.: Sandro Santos Andrade

1ª Avaliação Individual – 2016.2

Instruções:
- Todos os códigos-fonte produzidos, exceto arquivos produtos da compilação (ex: .class), devem ser enviados em um único arquivo .zip para sandro.andrade@gmail.com ao final da avaliação.
- O e-mail deve obrigatoriamente ter o subject: INF011–20162-P1.

Questão 1) (4,0) Implemente o N-ton Variante, variação do pattern Singleton que permite a criação de, no máximo, N diferentes instâncias de uma classe, sendo que M instâncias são criadas na inicialização da aplicação. A implementação deve ser realizada de modo que o código abaixo funcione conforme esperado.

```
Nton.initialize(3, 5);                  // No mínimo três e no máximo cinco instâncias
                                  // Três news já devem ocorrer nesta linha
Nton i1 = Nton.getInstance();             // Obtém a 1ª instância (sem new)
Nton i2 = Nton.getInstance();             // Obtém a 2ª instância (sem new)
Nton i3 = Nton.getInstance();             // Obtém a 3ª instância (sem new)
Nton i4 = Nton.getInstance();             // Obtém a 4ª instância (com new)
Nton i5 = Nton.getInstance();             // Obtém a 5ª instância (com new)
Nton i6 = Nton.getInstance();             // Obtém a 1ª instância
if (i6 == i1) System.out.println("OK");        // Deve exibir "OK"
Nton i7 = Nton.getInstance();             // Obtém a 2ª instância
if (i7 == i2) System.out.println("OK");        // Deve exibir "OK"
```

OBS: note que a partir da (N+1)ª instância, o sistema deve retornar as instâncias já criadas em formato de rodízio, ou seja, na 4ª invocação do *getInstance()* será retornada a 1ª instância, na 5ª a 2ª, na 6ª a 3ª, na 7ª a 1ª, e assim sucessivamente.

(1,0) – implementação correta do método initialize
(1,0) – implementação correta do armazenamento das múltiplas instâncias
(2,0) – implementação correta do método *getInstance*

Questão 2) (6,0) A principal desvantagem da Fábrica Flexível é a tipagem fraca do tipo do retorno do método *create* (IPrototype). Implemente uma fábrica flexível que usa um registro de Sub-Prototypes (classe base *IPrototype* e sub-prototypes *IMergeable* e *ISerializable*) para flexibilizar os produtos criados mantendo a tipagem. A implementação deve ser realizada de modo que o código abaixo funcione conforme esperado.

```
FlexibleFactory ff = new FlexibleFactory();
ff.addPrototype("Mergeable", "Pneu", new Pneu());
ff.addPrototype("Serializable", "Motor", new Motor());
Pneu p = (Pneu) ff.createMergeable("Pneu");
Motor m = (Motor) ff.createSerializable("Motor");
```

(2,0) – implementação correta da API de gerenciamento dos Prototypes
(2,0) – implementação correta do método *create*
(2,0) – implementação correta dos produtos

Boa sorte !