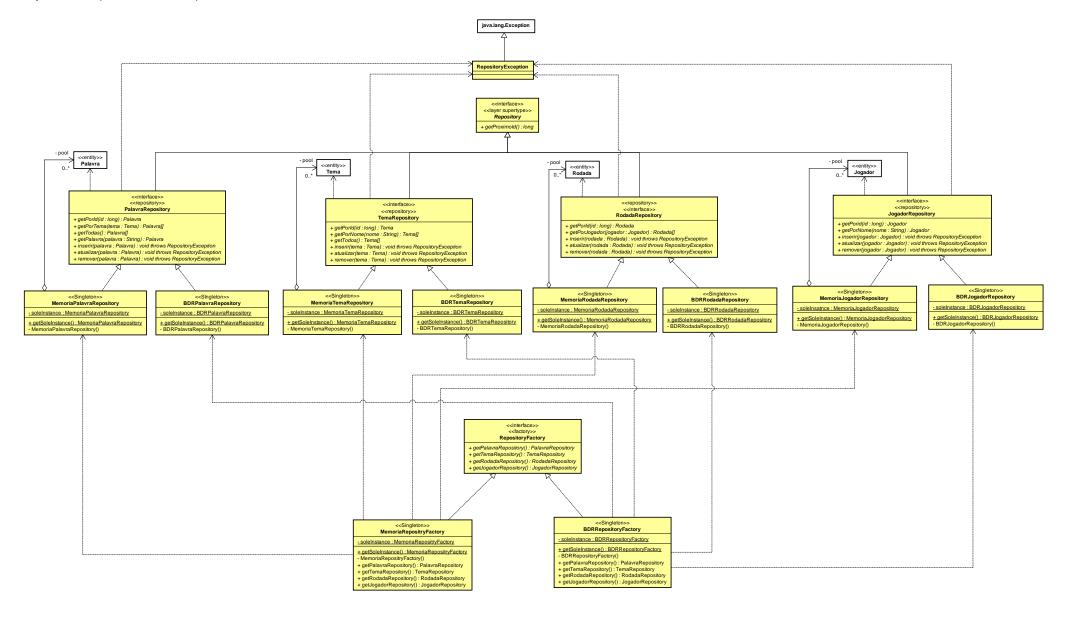
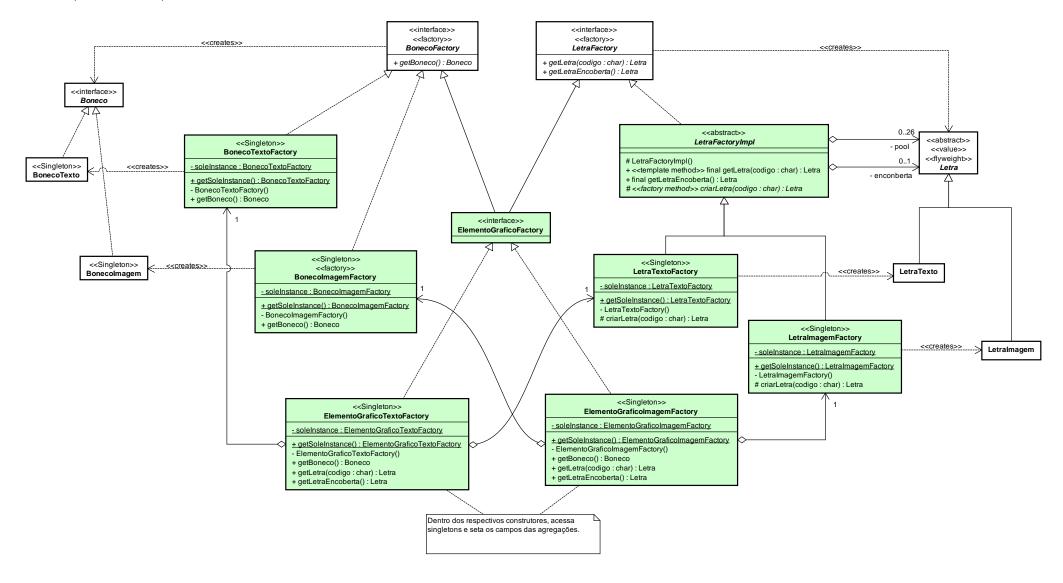
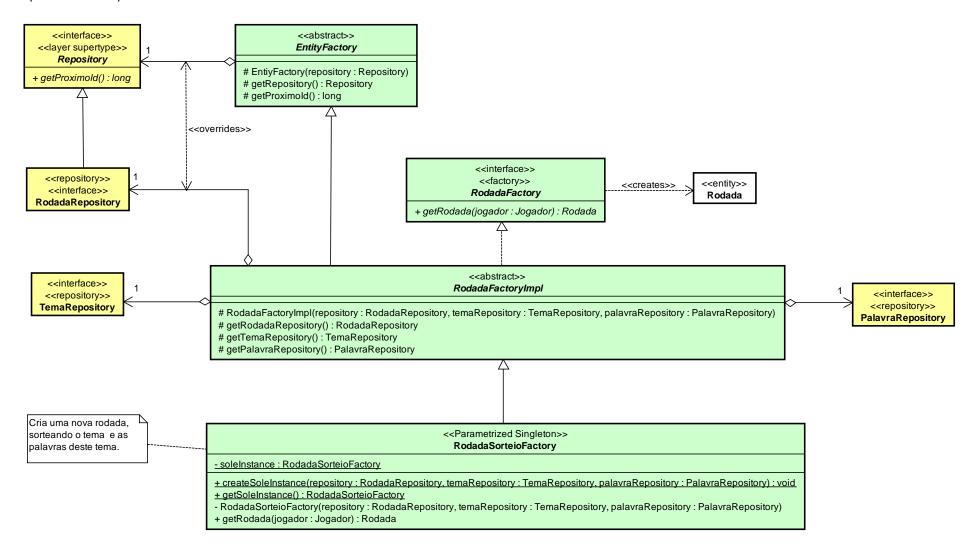
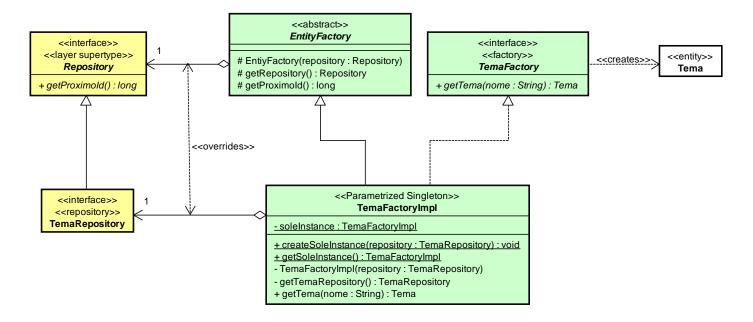


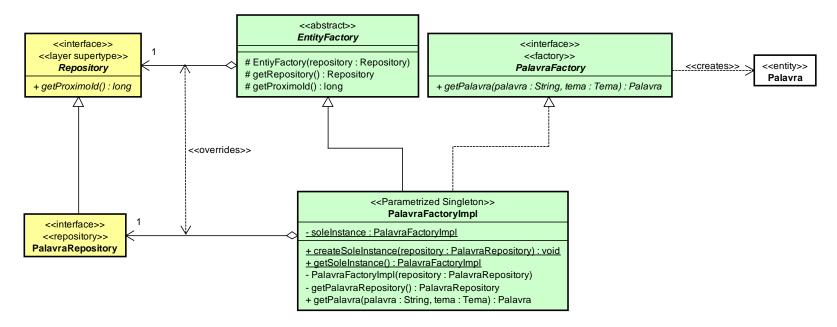
Repositórios (classe amarelas):

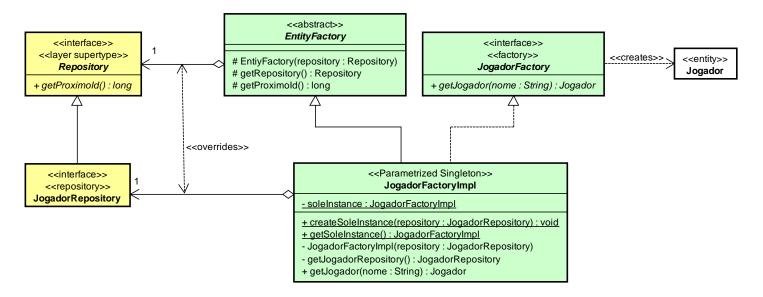




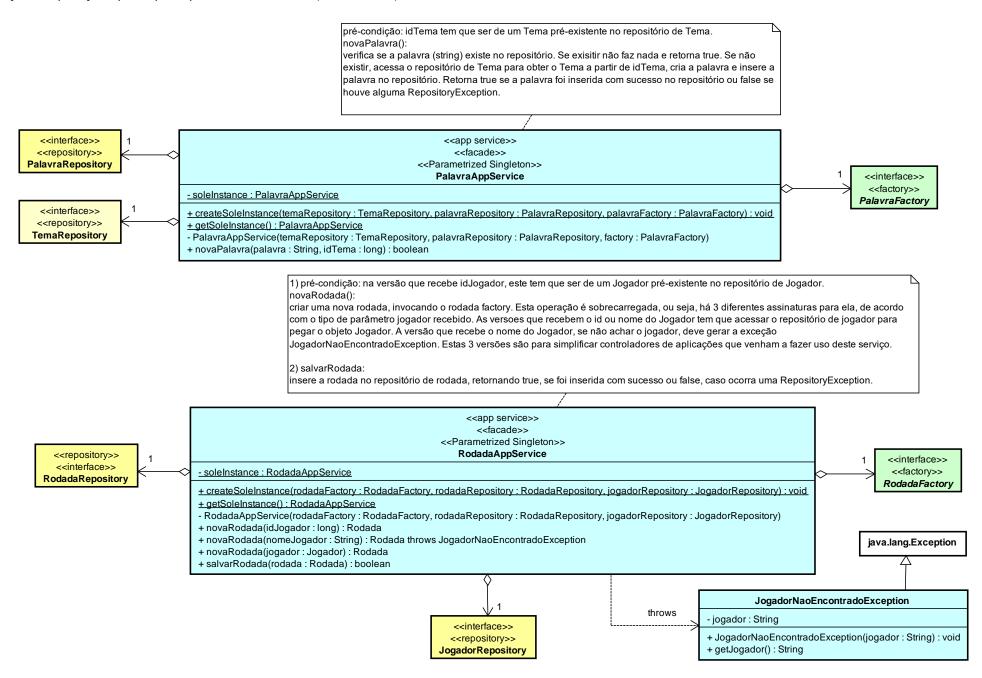








Serviços de aplicação - para apoiar passos de uses cases (classes azuis):



Classe de Configuração da Aplicação (classe roxa):

configurar():

- cria os singletons parametrizados da aplicação
- 2) Seta os factories estáticos de classes de domínio

Todas vez que alterar parâmetros desta classe, deve chamar configurar() para reconfigurar as classes da aplicação.

As operações públicas getBonecoFactory() e getLetraFactory() ambas são implementadas chamando a operação privada getElementoGraficoFactory(), assim:

return this.getElementoGraficoFactory();

<<Singleton>> <<pre><<pre><<pre>Aplicacao

- final TIPOS REPOSITORY FACTORY: string[] = {"memoria, relacional"}
- final TIPOS ELEMENTO GRAFICO FACTORY : string[] = {"texto", "imagem"}
- final TIPOS RODADA FACTORY: string[] = {"sorteio"}
- soleInstance : Aplicacao
- tipoRepositoryFactory : String = TIPOS_REPOSITORY_FACTORY[0]
- tipoElementoGraficoFactory : String = TIPOS_ELEMENTO_GRAFICO_FACTORY[0]
- tipoRodadaFactory : String = TIPOS_RODADA_FACTORY[0]
- + getSoleInstance(): Aplicacao
- Aplicacao()
- + configurar(): void
- + getTiposRepositoryFactory() : String[]
- + setTipoRepositoryFactory(tipo: String): void
- + getRepositoryFactory(): RepositoryFactory
- + getTiposElementoGraficoFactory(): String[]
- + setTipoElementoGraficoFactory(stipo : String) : void
- getElementoGraficoFactory(): ElementoGraficoFactory
- + getBonecoFactory(): BonecoFactory
- + getLetraFactory() : LetraFactory
- + getTiposRodadaFactory() : String[]
- + setTipoRodadaFactory(tipo : String) : void
- + getRodadaFactory(): RodadaFactory
- + getTemaFactory() : TemaFactory
- + getPalavraFactory(): PalavraFactory
- + getJogadorFactory() : JogadorFactory

Esta classe é um "Parametrized Factory". Dentro de seus métodos (getRepositoryFactory(), getElementoGraficoFactory() e getRodadaFactory()), define

os Singletons concretos a serem retornados, de acordo com tipos que foram setados, nos respectivos campos. É o único lugar da aplicação que conhece os Singletons concretos. Mas o retorno das operação são interfaces (abstrações), ou seja, desta classe para fora, todo o resto da aplicação, só conhece interfaces (abstrações), respeitando o principal princípio de OO: Open-Closed Principle (OCP).

Os métodos getTemaFactory(), getPalavraFactory() e getJogadorFactory(), simplesmente, retornam o respectivo Singleton concreto.