Alunos: BERNARDO FONSECA, JEAN SILVA

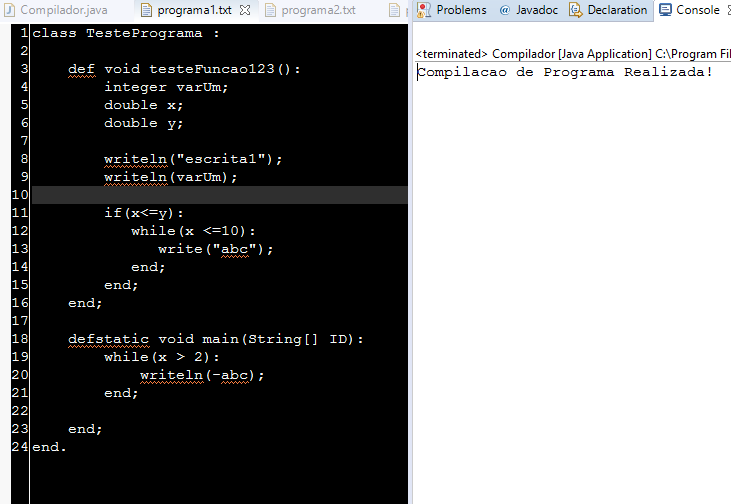
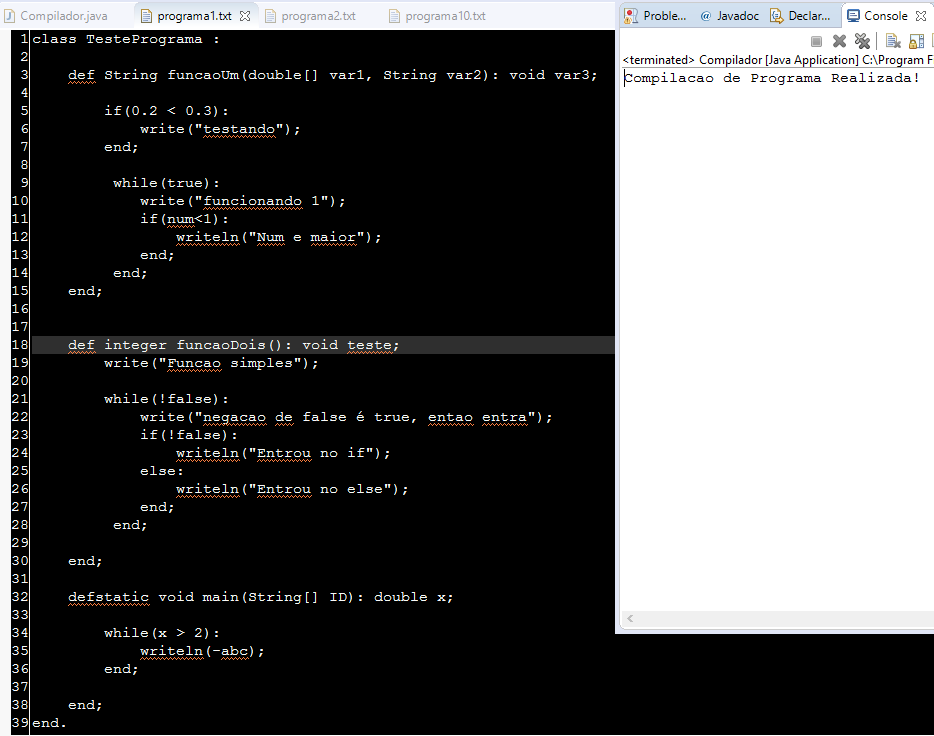
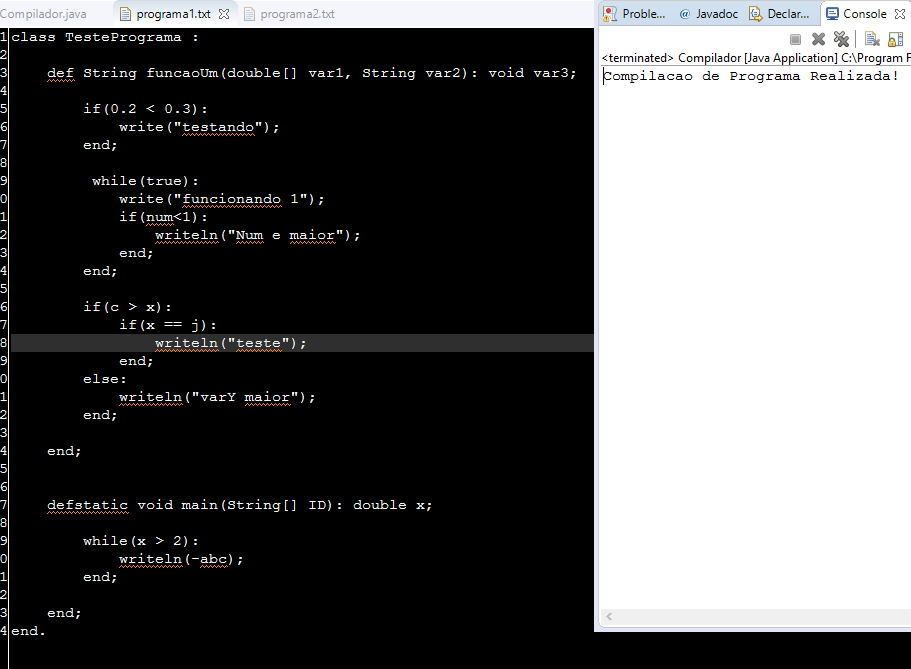
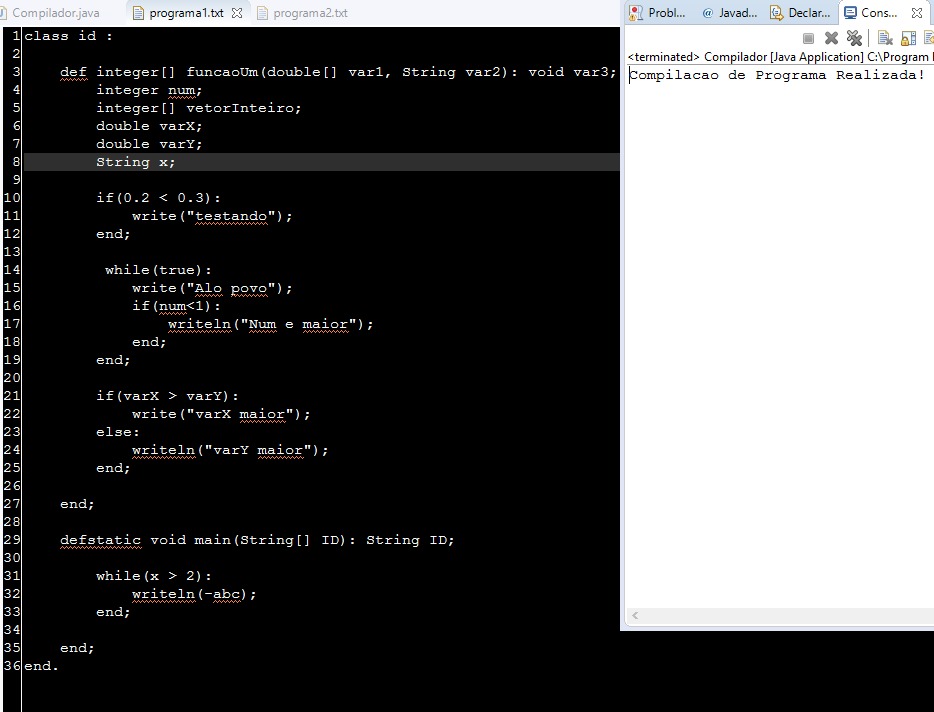
Enum NomeProducao: serviu para armazenar todos os nomes das produções, visando facilitar a programação e evitando erros. Quando era necessário passar algum nome de alguma produção por parâmetro, a utilização de constantes se torna mais facilitadora do que Strings, uma vez que Strings permitem erros de digitação, ao contrário de constantes enums em que se digitadas erradas, são reportados erros nas IDE’s.

Classe Producao: Foi utilizada esta classe com intuito de armazenar, para cada objeto Producao, seu nome e seu conjunto first e follow.

Classe Tabela: A classe tabela foi criada para armazenar todas as produções da linguagem. Ela possui uma estrutura HashMap e a chave para cada produção é seu respectivo método. É importante destacar que o HashMap é estático, possuindo assim uma mesma referência em qualquer parte do código, evitando-se ser instanciado multiplas vezes (evita gastos de memória). Sua única instância é criada dentro do parser através de um Singleton e assim que é criada, chama-se o método geraFirstFollow() no qual são atribuídos todos os Tokens Firsts e Follows de cada produção.

Classe Sintático: Nesta classe foi realizado todo o processamento da linguagem. Como citado anteriormente, o atributo tabela é instanciado no construtor da classe Sintático através do método Singleton.getInstance();. Desta forma, se ainda não existe uma tabela, esta é instanciada e chamado o método geraFirstFollow(). Caso exista, através do Singleton é retornado sua instância.  
O método isTokenEsperado(Tag classe) verifica se o token lido é da mesma classe da Tag passada por parâmetro e retorna true caso seja iguais ou false caso seja diferentes.  
O método isFirst(NomeProducao nome, Tag classe) retorna true caso exista um token com a classe passada por parâmetro no conjunto first da produção com nome passado por parâmetro.  
O método isFollow(NomeProducao nome, Tag classe) retorna true caso exista um token com a classe passada por parâmetro no conjunto follow da produção com nome passado por parâmetro.  
O método verificaProducaoVazia(NomeProducao nome) retorna true caso uma produção possua um token com a classa Tag.VAZIA e false caso contrário. Isso auxiliou na detecção de erros no modo pânico.

Funcionamento do parser: Foi utilizado uma estrutura de repetição while em praticamente todas as produção. Nele foi possível tomar decisões de acordo com os tokens reconhecidos pelo lexer. Dado um token de entrada, ele verifica através do isFirst() se o token identificado é um first da produção esperada. Caso seja, esta produção é chamada e o token é consumido (eat()) dentro dela. Caso não seja, o token é passado para o método isFollow() que verifica se ele é follow da produção esperada. Se ele for o follow da produção esperada, é realizada mais uma verificação para detectar se a **produção** esperada produz **cadeia vazia**. Produzindo cadeia vazia é sinal de que ela não precisa ser executada, ou seja, pode ser ignorada sem apresentar erro. Porém, se o token for o follow da produção esperada, mas a produção esperada não produz cadeia vazia, é sinal de que a produção não deveria ser “pulada” e é reportado um erro. O último caso verificado por esse loop é o caso do token não ser nem first e nem follow da produção esperada. Nesta situação, é reportado um erro, a variável que contabiliza a quantidade de erros é incrementada (errosSintaticos++) e o token é avançado pelo método advance();. Essa estrutura de repetição é realizada por todas as produções que chamam outras produções. Quando é alcançado o limite máximo de cinco erros, a execução da análise sintática é interrompida (System.exit) e é reportado uma mensagem de erro (System.err).

Abaixo, testes sintaticamente corretos:

Abaixo, testes sintaticamente incorretos:

