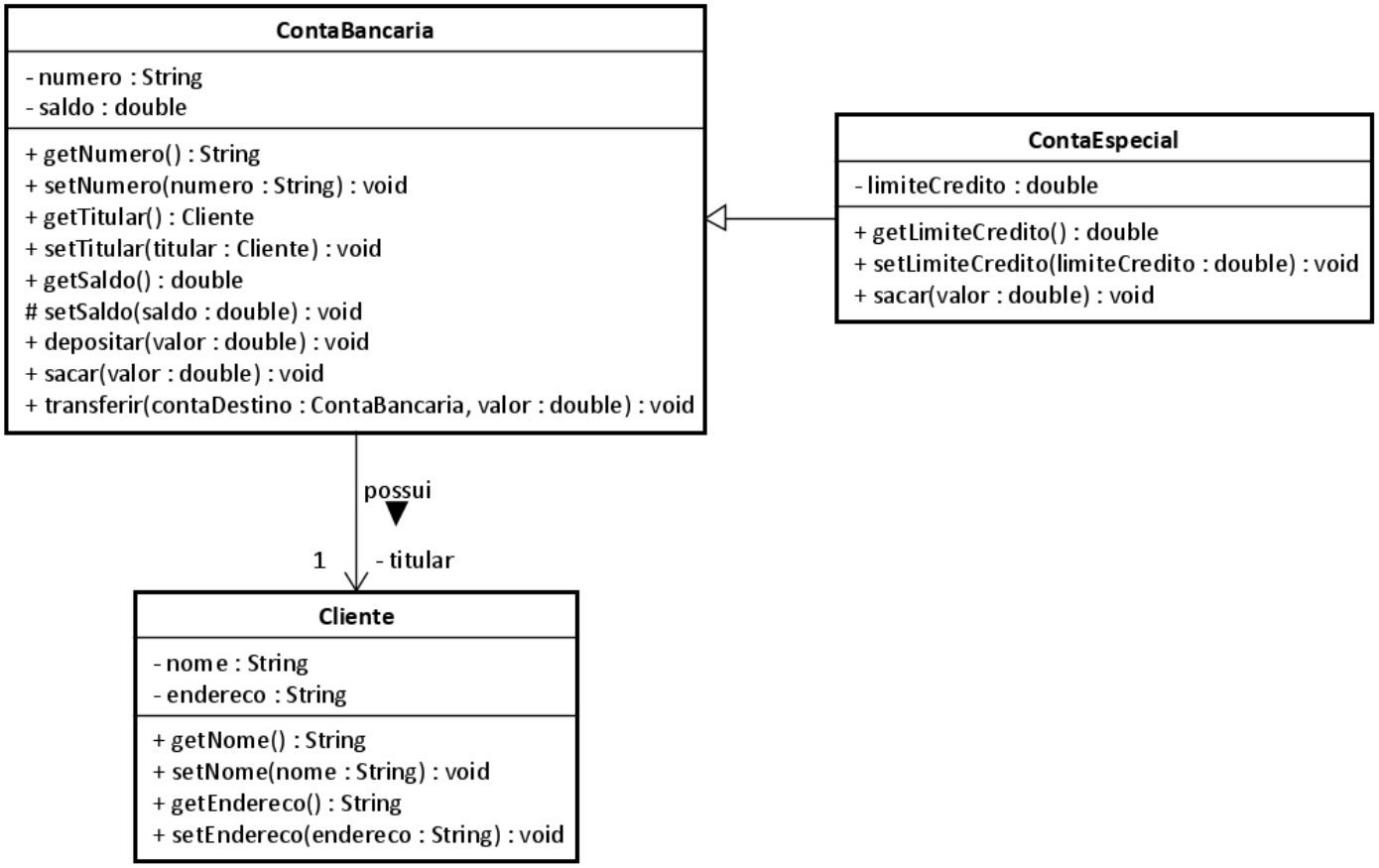
Exercício Herança

Crie um projeto novo para cada uma das questões abaixo.

Questão 1
Implemente o diagrama de classes abaixo.



A classe `ContaBancaria` deve implementar o método `setSaldo()`. Porém, este método não deve ser público e devemos dar oportunidade para que subclasses possam reutilizá-lo. Sendo assim, implemente-o usando o modificador de acesso `protected`, como consta no diagrama UML.

Já a classe `ContaEspecial` é uma conta bancária que permite utilizar um limite de crédito e com isso, deve aceitar que o cliente realize saques além do seu saldo, mas não superior ao seu limite de crédito. Como o método `sacar()` da superclasse não considera limite de crédito, precisamos *sobrescrever* este método na subclasse `ContaEspecial`.

Questão 2

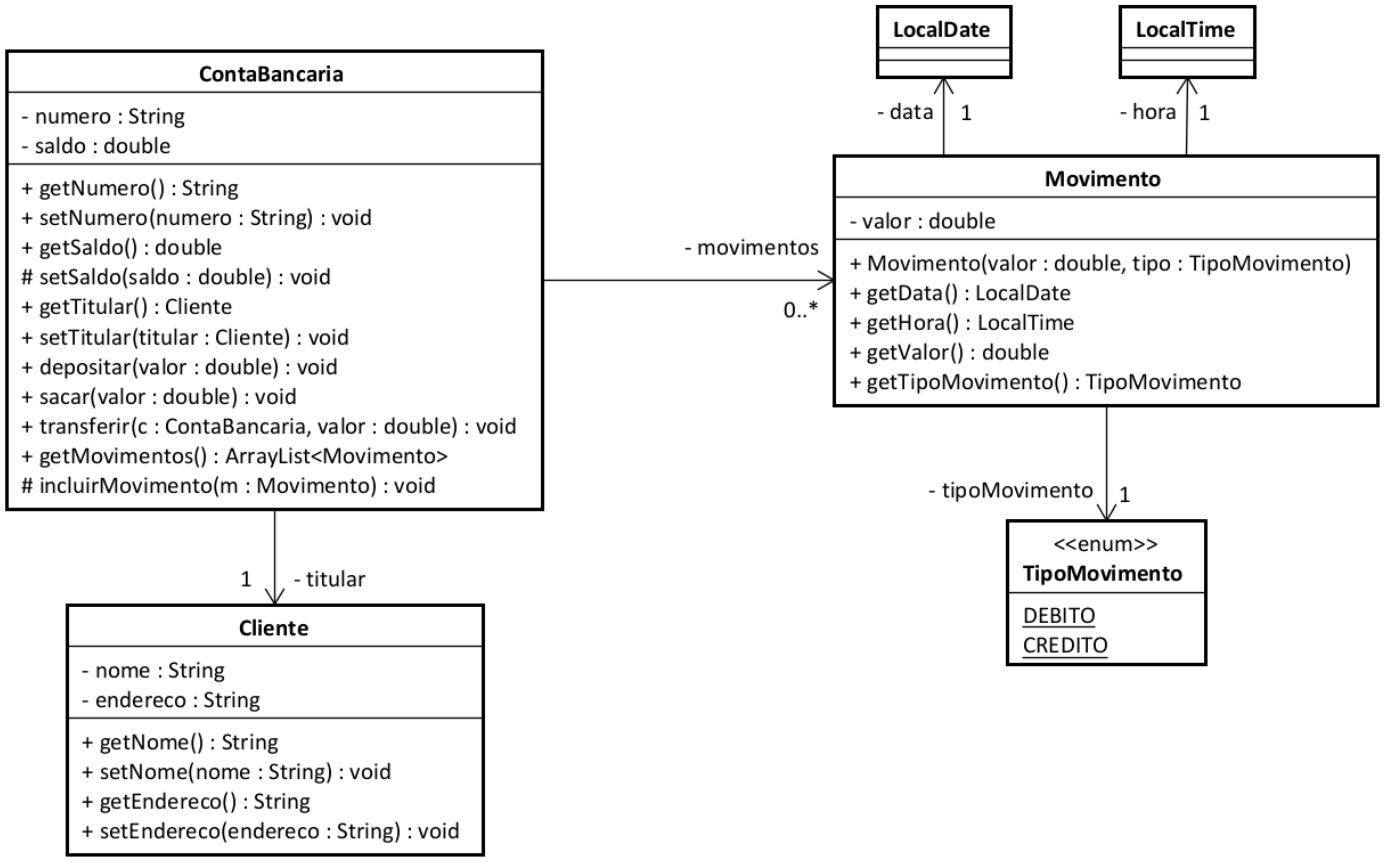
Implemente o plano de testes seguinte.

Plano de testes PL01 – Validar funcionamento da classe ContaEspecial			
Caso	Descrição	Entrada	Saída esperada
1	Conferir se o método sacar() permite sacar valor superior ao saldo	Criar uma conta bancária especial e definir seu limite de crédito em R$ 100,00. Realizar um depósito de R$ 20,00. Realizar um saque de R$ 50,00	O método getSaldo() deve resultar em -30,00.
2	Conferir se o método sacar() permite sacar valor superior ao saldo mas inferior ao limite	Criar uma conta bancária especial e definir seu limite de crédito em R$ 100,00.	O método getSaldo() deve resultar em -100,00.

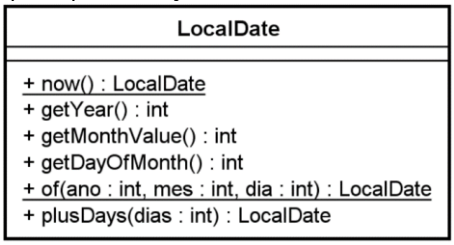
		Realizar um depósito de R$ 20,00. Realizar um saque de R$ 120,00	
3	Conferir se o método sacar() impede saques superiores ao saldo e limite de crédito	Criar uma conta bancária especial e definir seu limite de crédito em R$ 100,00. Realizar um depósito de R$ 20,00. Realizar um saque de R$ 120,01	Deve ser lançada uma exceção RuntimeException.

Questão 3

Copie as classes `ContaBancaria` e `Cliente` para um novo projeto. Modifique este projeto para que ele registre um histórico de todos os lançamentos de depósito e saque ocorridos na conta bancária, como visto no diagrama seguinte.

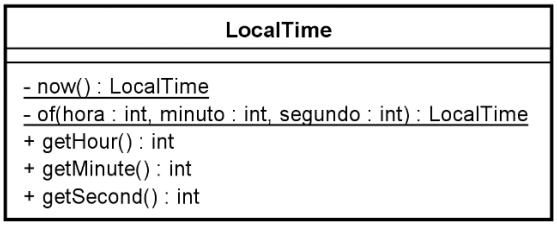


A classe `Movimento` é responsável por representar os movimentos de débito (saque) e crédito (depósito) que ocorreram numa conta bancária, constituindo assim, o histórico de movimentação de uma conta bancária. A ligação entre um objeto da classe `ContaBancaria` e um objeto da classe `Movimento` é feito através do método `depositar()`. Ou seja, quando o método `depositar()` for executado, deve criar um objeto da classe `Movimento` e adicioná-lo à `movimentos`. Da mesma forma, quando for feito um saque, o método `sacar()` deve criar um novo objeto da classe `Movimento` e compor `movimentos`. Quando as operações de débito e crédito forem executadas, deve-se considerar que o movimento ocorreu na data atual e na hora atual. Isto é, o construtor da classe `Movimento` deve automaticamente considerar que o movimento criado foi realizado na data e hora atuais. O objeto da classe `Movimento` deve refletir o tipo de operação (débito ou crédito) com o respectivo valor. `LocalDate` é uma classe da API Java que é utilizada para representar datas e encontra-se no pacote `java.time`. Veja o diagrama UML da classe apresentando seus principais serviços:



O método `now()` cria e retorna um objeto da classe `LocalDate`, que representa a data atual. O método `of()` cria e retorna um objeto da classe `LocalDate` que representa uma data específica, conforme argumentos fornecidos. Os métodos `getYear()`, `getMonthValue()` e `getDayOfMonth()` retornam, respectivamente, o ano, mês e dia da data representada pelo objeto da classe `LocalDate`.

Para representar horas, utilize a classe `LocalTime` que está contida na API Java no pacote `java.time`. Veja a seguir o diagrama com os métodos principais desta classe.



O método `now()` cria um objeto novo da classe `LocalTime`, que representa a hora atual. Os métodos `getHour()`, `getMinute()` e `getSecond()` retornam, respectivamente, a hora, minuto e segundo representados pelo objeto.

Questão 4

Crie um novo projeto no Netbeans e copie a implementação da questão 3 para este novo projeto. Em seguida, copie a classe `ContaEspecial` para este novo projeto também.

Em seguida, implemente o seguinte plano de testes:

Plano de testes PL01 – Validar funcionamento da classe ContaEspecial			
Caso	Descrição	Entrada	Saída esperada
1	Conferir se o método sacar gera movimento de crédito e débito na conta bancária	Criar uma conta bancária e depositar R$ 1000,00. Em seguida, sacar R$ 250,00.	getMovimentos() deve retornar uma lista com 2 elementos. O primeiro movimento gerado deve ser de crédito no valor de R$ 1000,00. Já o segundo elemento deve representar um movimento de débito no valor de R$ 250,00