Cenário principal	
Ações do ator	**Ações do sistema**
1. Selecionar opção Emitir Consumo por Período	
	2. Solicitar períodos desejados
3. Informar períodos	
	4. Selecionar todas as pizzas dos pedidos realizados no período
	5. Totalizar os ingredientes de cada sabor pedido
	6. Apresentar consumo no período solicitado
Restrições/validações	

17.2.3 Diagrama de Classes – Modelo de Domínio

Nesta seção, apresentaremos na figura 17.5 o modelo de domínio da solução para o sistema de pizzaria online.

Descreveremos as classes que compõem esse diagrama com suas associações, métodos e atributos. Embora já tenhamos afirmado isso no capítulo 4, sobre o diagrama de classes, acreditamos ser útil salientar novamente que, a rigor, deve haver um método `get` e um método `set` para cada atributo de uma classe. No entanto, essa modelagem enfoca o sistema em um nível mais alto. Assim, consideramos impraticável e mesmo desnecessário inserir um conjunto de métodos óbvios em classes que contenham um grande número de atributos. Dessa forma, definimos métodos como "registrar" ou "consultar" para inserir ou retornar conjuntos de atributos sempre que o processo não exigir métodos exclusivos para um atributo específico. Por esse motivo, muitos métodos aqui descritos retornam um atributo do tipo `String`, que poderá conter os valores de um conjunto de atributos necessários ao processo.

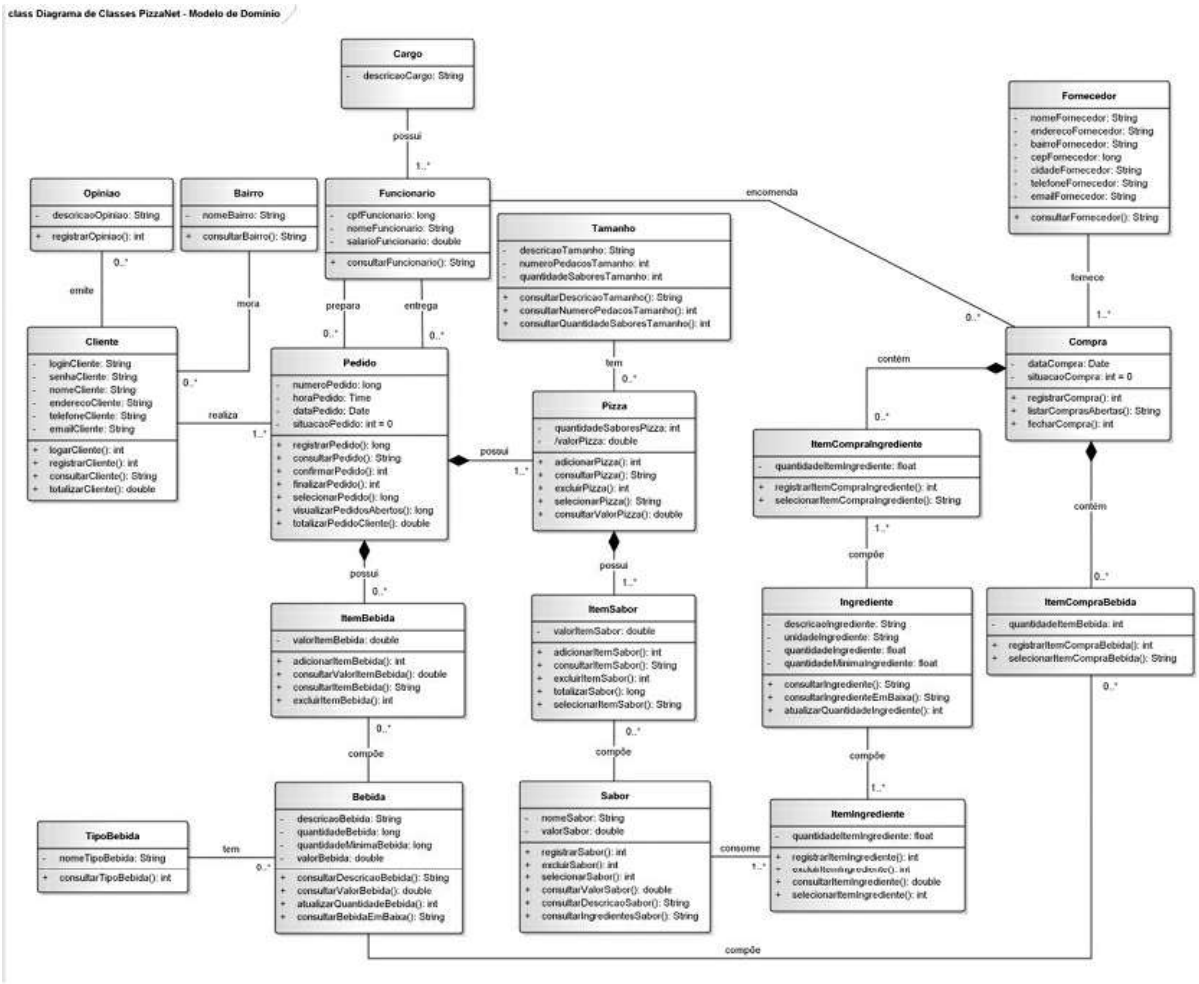

Figura 17.5 – Diagrama de Classes da PizzaNet – Modelo de Domínio.

- `Cliente` – Essa classe tem como função armazenar as informações relativas aos clientes da pizzaria, ou seja, pessoas que já solicitaram ao menos um pedido. Os atributos necessários a essa classe são `login`, `senha`, `nome`, `endereço`, `telefone` e `e-mail`, todos do tipo `String`. A classe contém, ainda, os seguintes métodos:

Método	Descrição
`logarCliente`	Permite autenticar um cliente na página do software, recebendo como parâmetros o login e a senha do cliente. Se estes estiverem corretos, o método retornará verdadeiro, informando que o cliente foi autenticado com sucesso, caso contrário, retornará falso.
`registrarCliente`	Utilizado para registrar um novo cliente. Recebe como parâmetros todos os atributos definidos na classe `Cliente` e retorna verdadeiro, se o cliente foi registrado com sucesso, ou falso, caso contrário.
	Permite pesquisar um determinado cliente. Esse método retorna uma

`consultarClienteString` contendo o nome do cliente consultado.

`totalizarCliente` Possui como objetivo totalizar todos os pedidos já realizados por um cliente para determinar quais os melhores clientes da pizzaria. O método retorna um `double` contendo os valores totais gastos por um cliente. É utilizado em conjunto com outros métodos, definidos em outras classes, como `totalizarPedido`, que serão explicados ao longo desta seção.

- `Opiniao` – Essa classe armazena as opiniões registradas pelos clientes relativas a seu nível de satisfação quanto aos pedidos solicitados. Tem como únicos atributos a descrição da opinião fornecida, do tipo `String`, e o método `registrarOpiniao`, que instancia um novo objeto da classe, retornando verdadeiro quando é bem-sucedido. Observe que um objeto da classe `Opiniao` está vinculado a somente um objeto da classe `Cliente`, mas um objeto dessa última classe pode estar vinculado a muitas instâncias da classe `Opiniao`.
- `Bairro` – Essa classe contém as informações relativas aos bairros onde estão localizados os clientes da pizzaria. Seu único atributo é o nome do bairro, do tipo `String`. Aqui, identificamos somente o método `consultarBairro`, usado para consultar o bairro do cliente, retornando uma `String` com o nome do bairro consultado. Obviamente, é necessário existir um método para registrar novos bairros, no entanto, uma vez que o diagrama tem um conjunto muito grande de informações, procuramos definir somente os métodos realmente importantes. Muitos clientes podem morar em um determinado bairro, como demonstra a associação **mora**, mas um cliente específico só pode morar em um único bairro.
- `Cargo` – Essa classe armazena os cargos assumidos pelos funcionários da empresa. Seu único atributo é a descrição do cargo do tipo `String`.
- `Funcionario` – A classe `Funcionario` armazena as informações dos funcionários que trabalham ou trabalharam na empresa. Seus atributos contêm o CPF do funcionário, do tipo `long`, o nome, do tipo `String`, e o salário, do tipo `double`. O único método aqui descrito é o `consultarFuncionario`, que serve para consultar um funcionário, retornando uma `String` contendo seu nome. Observe que essa classe

tem uma associação com a classe anterior, que determina que um funcionário deve ter um único cargo, mas um mesmo cargo pode ser atribuído a muitos funcionários.

- `Pedido` – Essa classe armazena as informações relacionadas aos pedidos solicitados pelos clientes. Observe que a classe tem duas associações com a classe `Funcionario`, na qual um funcionário pode preparar e/ou entregar muitos pedidos, mas um pedido só pode ser preparado por um funcionário específico e, da mesma forma, só pode ser entregue por um determinado funcionário (não necessariamente o mesmo). A classe `Pedido` tem também uma associação com a classe `Cliente`, que determina que um cliente pode solicitar muitos pedidos, mas um pedido deve estar associado somente a um cliente. Há, ainda, outras associações dessa classe com outras classes, que serão explicadas ao longo desta seção.

A classe `Pedido` contém os atributos número do pedido, do tipo `long`, a hora em que o pedido foi solicitado, do tipo `Time`, a data em que este foi realizado, do tipo `Date`, e a situação do pedido, do tipo `int`, que determina se o pedido ainda não foi atendido, se já foi finalizado ou já foi entregue. É preciso destacar que os tipos `Time` e `Date` referem-se a classes que devem ser implementadas pela linguagem ou estar contidas no sistema. Essa classe tem ainda os seguintes métodos:

- `Tamanho` – Essa classe armazena os tamanhos disponibilizados pela PizzaNet. Seus atributos são a descrição do tamanho, do tipo `String`, o número de pedaços e a quantidade de sabores, ambos do tipo `int`. Essa classe tem ainda os seguintes métodos:

Método	Descrição
`consultarDescricaoTamanho`	Retorna uma `String` contendo a descrição do tamanho (se é pequeno, médio ou grande).
`consultarNumeroPedacosTamanho`	Retorna um inteiro contendo o número de pedaços do tamanho.
`consultarQuantidadeSaboresTamanho`	Retorna um inteiro contendo o número de sabores que uma pizza desse tamanho pode conter.

- `Pizza` – Essa classe armazena as informações das pizzas solicitadas em um determinado pedido. Observe que existe uma associação de

composição entre a classe `Pedido` e a classe `Pizza`, o que significa que os objetos da classe `Pedido` são `objetos-todo` cujas informações devem ser complementadas pelas informações contidas em `objetos-parte` da classe `Pizza` (essas informações devem ser complementadas também por `objetos-parte` da classe `ItemBebida`, que será explicada mais adiante). Dessa maneira, como podemos observar pela associação de composição entre essas duas classes, um pedido pode conter muitas pizzas, mas uma pizza deve estar contida única e exclusivamente em um pedido.

É preciso notar também que existe uma associação entre essa classe e a classe tamanho, determinando que uma pizza pode estar associada a somente um tamanho, mas um tamanho pode estar relacionado a muitas pizzas. Existe ainda outra associação que será explicada quando abordarmos a próxima classe.

A classe `Pizza` contém os atributos quantidade de sabores, do tipo `int`, e valor da pizza, do tipo `double`. O leitor poderá se perguntar o porquê do atributo quantidade de sabores, uma vez que essa informação pode ser puxada da classe `Tamanho` relacionada à classe `Pizza`. Isso estaria correto, mas pode acontecer de um cliente solicitar uma pizza grande com um único sabor. Por esse motivo, optamos por inserir esse atributo nessa classe. O leitor poderá inquirir também por que existe o atributo valor da pizza, uma vez que este é calculado, como demonstra a barra ao lado de sua visibilidade. Optamos por inserir esse atributo para facilitar os processos que pesquisarão as pizzas dos pedidos, diminuindo o número de operações a ser realizadas e as classes a ser consultadas.

Essa classe contém ainda os seguintes métodos:

Método	Descrição
`adicionarPizza`	Serve para adicionar uma pizza ao pedido. Retorna um inteiro que determina se o método foi executado com sucesso ou não.
`consultarPizza`	Permite consultar uma pizza de um pedido e retorna uma `String` contendo os dados da pizza consultada.
`excluirPizza`	Por meio desse método é possível excluir uma pizza de um pedido. O método retorna um inteiro que conterá um valor verdadeiro (1), caso o método consiga excluir a pizza, ou falso (0), caso contrário.
`selecionarPizza`	Seleciona uma pizzaa de um pedido, retornando uma `String` contendo os totais dos ingredientes consumidos em cada pizza.
	Permite consultar o valor de uma pizza específica, retornando um

`consultarValorPizza`double contendo seu valor.

- `ItemSabor` – Essa classe armazena as informações sobre os sabores escolhidos para uma pizza. Observe que existe uma associação de composição entre a classe `Pizza` e a classe `ItemSabor`. Isso significa que os objetos da classe `Pizza` são também `objetos-todo` em relação aos objetos da classe `ItemSabor`, que desempenham o papel de `objetos-parte` nessa associação, uma vez que estes devem complementar as informações de um objeto da classe `Pizza`, fornecendo os dados de cada sabor escolhido para uma pizza específica.

Essa classe contém o atributo `valorItemSabor`, do tipo `double`. Alguém poderia se perguntar se esse atributo é realmente necessário, uma vez que pode ser acessado a partir da classe `Sabor`, que explicaremos a seguir. Esse atributo é realmente necessário porque o valor de um sabor pode ser alterado, o que deturparia as informações relativas ao valor do sabor na época em que o pedido foi feito.

A classe `ItemSabor` contém ainda o seguintes métodos:

Método	Descrição
`adicionarItemSabor`	Permite adicionar um sabor a uma pizza. Retorna um inteiro determinando o sucesso ou não do método.
`consultarItemSabor`	Consulta um sabor específico de uma pizza, retornando uma `String` contendo seus dados.
`excluirItemSabor`	Permite excluir um sabor de uma pizza, retornando um inteiro que determina o sucesso ou não dessa operação.
`totalizarSabor`	Utilizado para totalizar quantas vezes um sabor foi pedido. O método retorna um `long` contendo esse total.
`selecionarItemSabor`	Chamado pelo método `selecionarPizza` para selecionar todos os objetos da classe `ItemSabor` associados à pizza. O método retorna uma `String` com os dados desses objetos.

- `Sabor` – Essa classe armazena as informações relacionadas aos sabores oferecidos pela pizzaria. Seus atributos são o nome do sabor, do tipo `String`, e o valor, do tipo `double`. Observe que essa classe tem uma associação com a classe `ItemSabor`, que determina que um objeto da classe `Sabor` pode estar associado a muitos objetos da classe `ItemSabor`, mas um objeto da classe `ItemSabor` só pode estar relacionado a um

objeto da classe `Sabor`. A classe contém ainda os métodos:

Método	Descrição
`registrarSabor`	Permite o registro de um novo sabor. Retorna um inteiro que, se contiver verdadeiro (1), indicará que foi possível registrar o novo sabor, caso contrário, determinará que algum erro ocorreu.
`excluirSabor`	Permite a exclusão de um sabor. Retorna um inteiro para determinar o sucesso ou não da operação.
`selecionarSabor`	Usado durante o processo de conclusão de um pedido para auxiliar a dar baixa nos ingredientes utilizados na produção dos sabores. Sua função é selecionar todos os sabores solicitados no pedido. O método retorna um inteiro determinando se o método foi concluído com sucesso ou não.
`consultarValorSabor`	Permite consultar o valor de um sabor, retornando um `double` contendo seu valor.
`consultarDescricaoSabor`	Permite consultar a descrição do sabor, retornando uma `String` contendo sua descrição.
`consultarIngredientesSabor`	Permite consultar os ingredientes necessários ao preparo de um sabor. Retorna uma `String` contendo a listagem dos ingredientes necessários.

O leitor talvez se pergunte por que é necessária a classe `ItemSabor` e se sua função não poderia ser assumida pela classe `Sabor`. Ocorre, porém, que uma pizza pode ter muitos sabores e um sabor pode estar contido em muitas pizzas. Sendo assim, é necessário haver uma classe intermediária entre as classes `Pizza` e `Sabor` que armazene as informações dos sabores de uma pizza específica.

- `Ingrediente` – Essa classe armazena as informações referentes aos ingredientes necessários para preparar os sabores oferecidos pela PizzaNet. Seus atributos são a descrição do ingrediente, a unidade do ingrediente, ambos do tipo `String`, além da quantidade em estoque e a quantidade mínima do produto (se a quantidade do ingrediente estiver igual ou abaixo do mínimo, deverá ser feita solicitação de compra a um fornecedor), ambos do tipo `double`.

Método	Descrição
`consultarIngrediente`	Permite consultar um ingrediente específico, retornando uma `String` com sua descrição.
	Retorna uma String contendo as informações de

`consultarIngredienteEmBaixa`	todos os ingredientes cuja quantidade em estoque estiver igual ou abaixo do mínimo.
`atualizarQuantidadeIngrediente`	Diminui a quantidade necessária à produção de um sabor da quantidade em estoque de um ingrediente. Retorna um inteiro que determina o sucesso ou não da operação.

- `ItemIngrediente` – Esta é uma classe intermediária que contém as informações de cada ingrediente necessário à produção de um sabor. Essa classe justifica-se uma vez que um sabor pode estar relacionado a muitos ingredientes e um ingrediente pode estar associado a muitos sabores. Assim, cada objeto dessa classe está associado a um objeto da classe `Ingrediente` e a um objeto da classe `Sabor`, mas um objeto da classe `Sabor` pode estar associado a muitos objetos da classe `ItemIngrediente`, o mesmo ocorrendo em relação aos objetos da classe `Ingrediente`.

Essa classe armazena o atributo quantidade do tipo `double`, que estabelece a quantidade necessária de um ingrediente específico para um sabor específico. A classe tem ainda os seguintes métodos:

Método	Descrição
`registrarItemIngrediente`	Permite registrar um novo item de ingrediente.
`excluirItemIngrediente`	Permite excluir um item de ingrediente.
`consultarItemIngrediente`	Permite consultar um item de ingrediente, retornando um `double` contendo sua quantidade.
`selecionarItemIngrediente`	Permite selecionar os itens de ingrediente de um determinado sabor.

- `TipoBebida` – Essa classe armazena os tipos de bebida oferecidos pela PizzaNet, ou seja, suco, refrigerante e cerveja. Seu único atributo é o nome, do tipo `String`, e seu único método, `consultarTipoBebida`, permite consultar um tipo de bebida, retornando uma `String` contendo seu nome.
- `Bebida` – Essa classe armazena os dados de todas as bebidas comercializadas pelo sistema. Note que esta classe está associada à classe `TipoBebida`. A multiplicidade dessa associação determina que uma bebida deve estar associada a somente um tipo de bebida, no entanto um tipo de bebida pode estar associado a muitas bebidas. Os atributos dessa classe são a descrição da bebida, do tipo `String`, a quantidade em

estoque e a quantidade mínima da bebida, ambos do tipo `long`, e o valor da bebida, do tipo `double`.

Essa classe contém os seguintes métodos:

Método	Descrição
`consultarDescricaoBebida`	Permite consultar a descrição de uma bebida, retornando uma `String` contendo a descrição.
`consultarValorBebida`	Permite consultar o valor de uma bebida, retornando um `double` contendo o valor em questão.
`atualizarQuantidadeBebida`	Diminui a quantidade solicitada em um pedido da quantidade em estoque de uma bebida. Retorna verdadeiro, se a operação pôde ser realizada, e falso, caso contrário.
`consultarBebidaEmBaixa`	Consulta todas as bebidas com a quantidade em estoque abaixo da quantidade mínima, retornando uma `String` com os dados de cada bebida em baixa.

- `ItemBebida` – Essa é uma classe intermediária posicionada entre a classe `Pedido` e a classe `Bebida`. É necessária para armazenar as bebidas solicitadas em um pedido específico, pelo fato de que um pedido pode incluir muitas bebidas e uma bebida pode estar presente em muitos pedidos, não havendo espaço em nenhuma das duas classes para armazenar as informações das bebidas do pedido, já que não é possível saber quantas bebidas terá um pedido, nem quantos pedidos solicitarão uma bebida. Mesmo que se reservasse um espaço grande em qualquer das duas classes, este poderia ser atingido e, enquanto isso não ocorresse, um grande número de atributos seria deixado em branco.

Observe que a classe `Pedido` tem uma associação de composição com a classe `ItemBebida`. Isso significa que as informações de um `objeto-todo` da classe `Pedido` precisam ser complementadas pelas informações contidas nos `objetos-parte` da classe `ItemBebida`. Além disso, essa associação nos passa a informação de que um objeto `ItemBebida` está associado a um único objeto da classe `Pedido` e só deverá ser excluído também se o pedido a que estiver associado for destruído.

A classe `ItemBebida` também está associada à classe `Bebida` e a multiplicidade dessa associação passa a informação de que uma bebida pode estar associada a muitos itens de bebida, mas um item de bebida tem que estar associado a somente uma bebida.

Essa classe tem como atributo somente o valor do item de bebida do tipo `double`. A exemplo da classe `ItemSabor`, armazenamos o valor da bebida na época em que o pedido foi feito, já que o valor armazenado na classe `Bebida` pode ser atualizado. Assim, se puxássemos o valor da classe bebida, o total apresentado em um pedido consultado poderia ser diferente do real, se tivesse ocorrido algum aumento no valor da bebida.

A classe `ItemBebida` contém ainda os métodos:

Método	Descrição
`adicionarItemBebida`	Permite adicionar uma nova bebida a um pedido, retornando verdadeiro, se o método foi executado com sucesso, ou falso, caso contrário.
`consultarValorBebida`	Consulta o valor de uma bebida, retornando um `double` contendo seu valor.
`consultarItemBebida`	Consulta um item de bebida, retornando uma `String` com seus dados. É usado em conjunto com o método `consultarDescricaoBebida`, já explicado na classe anterior.
`excluirItemBebida`	Permite excluir um item de bebida do pedido. O método retorna verdadeiro (1), se for finalizado com sucesso, ou falso (0), caso contrário.

- `Fornecedor` – Essa classe armazena as informações dos fornecedores dos quais a pizzaria compra ingredientes e bebidas. Seus atributos são nome, endereço, bairro, cidade, telefone e e-mail do tipo `String`, além do `CEP` do tipo `long`. A classe contém, ainda, o método `consultarFornecedor` que permite consultar um fornecedor, retornando uma `String` com seus dados.
- `Compra` – Essa classe contém as informações relativas aos pedidos de compras solicitados aos fornecedores. A classe contém os atributos data, do tipo `Date` (que se refere a uma classe), que armazena a data em que a compra foi solicitada, e situação, do tipo `int`, que armazena 0 (seu valor inicial), caso a compra ainda não tenha sido entregue, ou 1, caso contrário.

Essa classe tem uma associação com a classe `Fornecedor`, que informa que a um fornecedor podem ter sido solicitadas muitas compras, mas uma compra está associada a somente um fornecedor. Há também uma associação com a classe `Funcionario`, que determina que um

funcionário pode encomendar muitas compras, mas uma compra só pode estar associada a um funcionário. Há, ainda, outras associações que serão explicadas quando da explanação de outras classes.

Essa classe contém os seguintes métodos:

Método	Descrição
`registrarCompra`	Permite registrar uma nova compra, retornando 1 (verdadeiro), se for possível registrar a nova compra, ou falso (0), caso contrário.
`listarComprasAbertas`	Retorna uma `String` contendo os dados de todas as compras ainda não entregues pelos fornecedores.
`fecharCompra`	Permite definir uma compra como concluída, ou seja, uma compra que foi entregue pelo fornecedor, de acordo com o que foi solicitado. Esse método muda o valor do atributo `situacao` para 1, significando que a compra está fechada.

- `ItemCompraIngrediente` – Esta é uma classe intermediária que armazena as informações dos ingredientes solicitados em uma determinada compra. Observe que existe uma associação de composição entre a classe `Compra` e a classe `ItemCompraIngrediente`, determinando que um objeto da classe `Compra` é um `objeto-todo` cujas informações precisam ser complementadas pelos `objetos-parte` da classe `ItemCompraIngrediente`. Assim, um objeto dessa classe precisa estar associado a um único objeto da classe `Compra`, embora este possa estar associado a muitos objetos da classe `ItemCompraIngrediente`.

Há, ainda, uma associação dessa classe com a classe `Ingrediente`, que determina que um objeto da classe `Ingrediente` pode estar associado a muitos objetos da classe `ItemCompraIngrediente`, mas um objeto da classe `ItemCompraIngrediente` precisa estar vinculado, nessa associação, a somente um objeto da classe `Ingrediente`.

Essa classe tem como atributo unicamente a quantidade do ingrediente solicitada em cada compra, do tipo `double`. Além disso, apresenta ainda os seguintes métodos:

Método	Descrição
`registrarItemCompraIngrediente`	Permite instanciar um novo objeto dessa classe, retornando verdadeiro, se for executado com sucesso, ou falso, caso contrário.
`selecionarItemCompraIngrediente`	Seleciona cada ingrediente solicitado em uma determinada compra, retornando uma `String`

com seus dados.

- `ItemCompraBebida` – Esta é também uma classe intermediária, posicionada entre as classes `Compra` e `Bebida`, com uma função bastante semelhante à classe `ItemCompraIngrediente`. Essa classe armazena as informações das bebidas solicitadas em uma determinada compra. O leitor notará que existe também uma associação de composição entre a classe `Compra` e a classe `ItemCompraBebida`, o que significa que as informações dos `objetos-parte` dessa classe devem também complementar as informações dos `objetos-todo` da classe `Compra`. Essa associação também informa que um objeto da classe `ItemCompraBebida` necessita estar vinculado, nessa associação, a somente um objeto da classe `Compra`, mas um objeto da classe `Compra` pode estar associado a muitos objetos da classe `ItemCompraBebida`.

  Essa classe tem também uma associação com a classe `Bebida` e tal associação informa que um objeto da classe `Bebida` pode estar associado a muitos objetos da classe `ItemCompraBebida`, mas um objeto da classe `ItemCompraBebida` precisa estar vinculado, nessa associação, a somente um objeto da classe `Bebida`.

  Essa classe tem como atributo unicamente a quantidade da bebida solicitada em cada compra, do tipo `double`. Além disso, contém ainda os seguintes métodos:

Método	Descrição
`registrarItemCompraBebida`	Permite instanciar um novo objeto dessa classe, retornando verdadeiro, se for executado com sucesso, ou falso, caso contrário.
`selecionarItemCompraBebida`	Seleciona cada bebida solicitada em uma determinada compra, retornando uma `String` com seus dados.

17.2.4 Diagrama de Objetos

Nesta seção apresentaremos, na figura 17.6, um diagrama de objetos referente ao modelo de domínio apresentado na seção 17.2.3. O objetivo desse diagrama é meramente ilustrativo, com o intuito de aclarar como estarão organizados os objetos do sistema quando instanciados, como, aliás, é o objetivo de todo diagrama de objetos.