**Atividade Prática 1 – APPOO Prof. Gabriela Nunes Lopes 2025/2**

**Alunos: Otávio Serafim de Souza Matos/Germano Marques Cipriano Fagundes**

**1 (Aula 05)** - Faça o diagrama UML e implemente em Python. Coloque aqui os prints dos seus códigos e dos resultados dos testes. Implemente as classes e main em .py separados.

Crie uma classe bolo;

Crie uma instância da classe bolo;

Crie dois tipos de bolo com os seguintes atributos:

Nome

Recheio

Massa

Se tem cobertura – o primeiro tem e o segundo não

Crie um terceiro bolo, igual ao primeiro, mas sem cobertura:

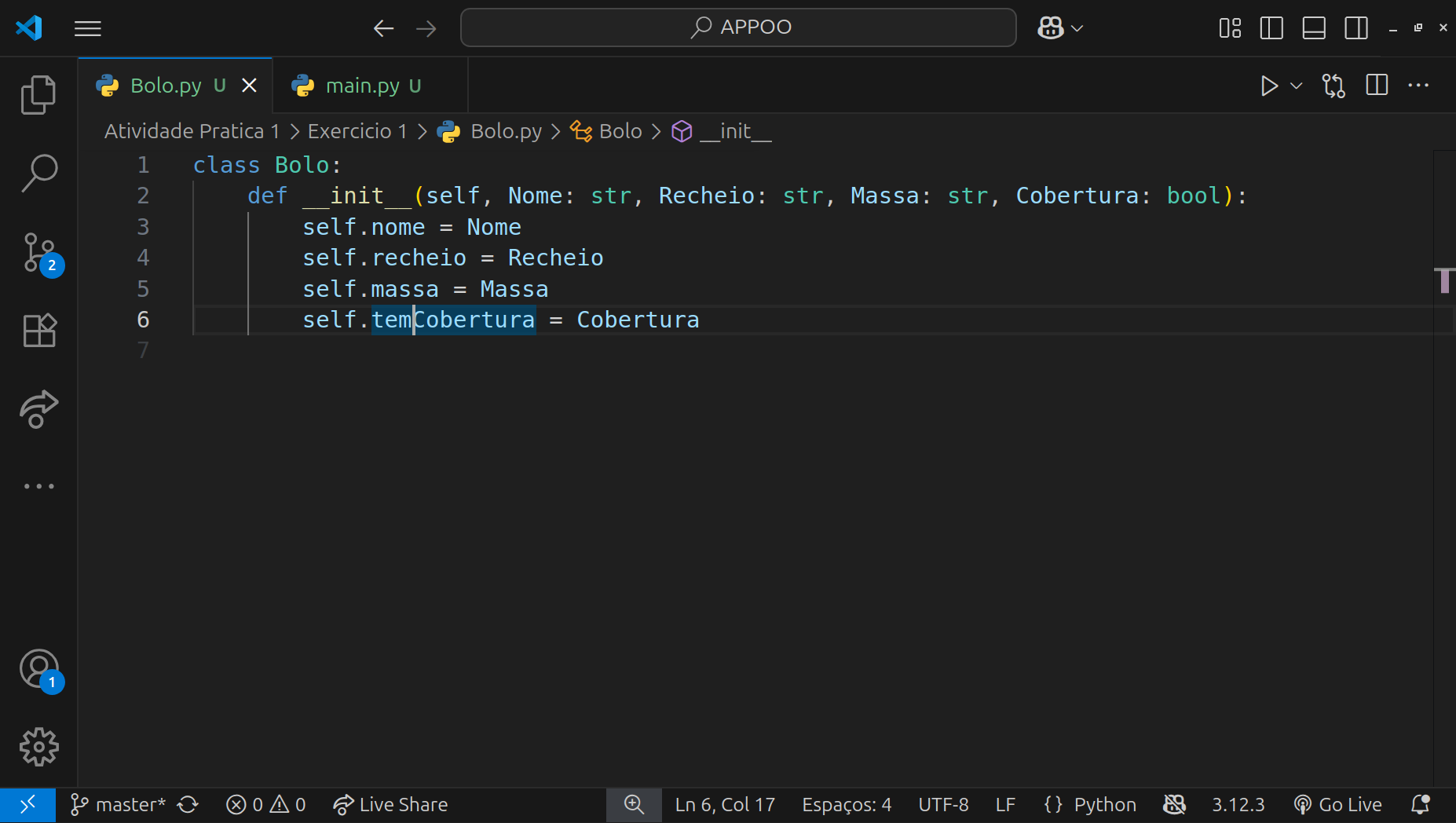
Advindo da instância principal (crie a partir da classe com os atributos requeridos)

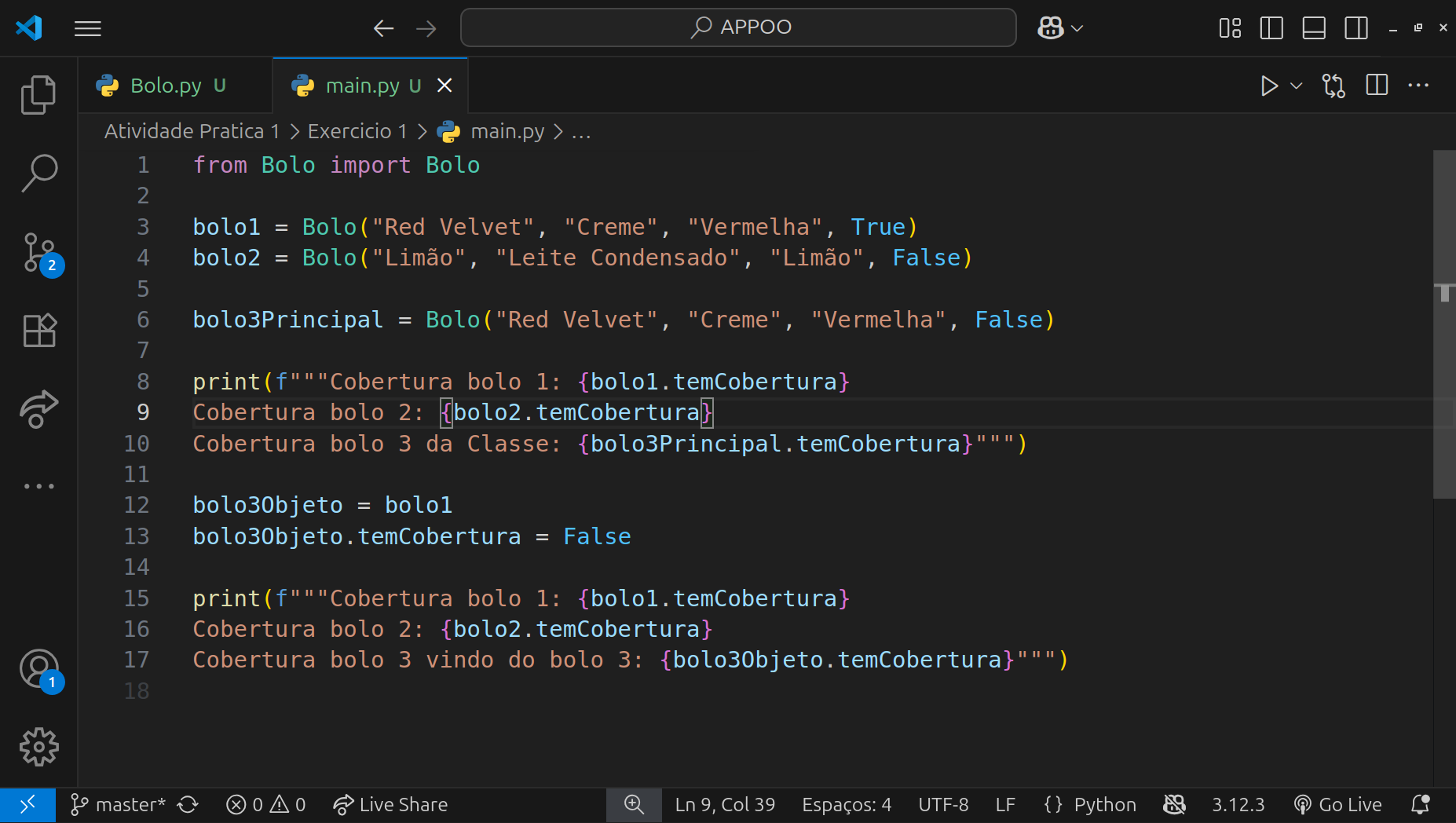
Printe na tela a cobertura de todos os bolos

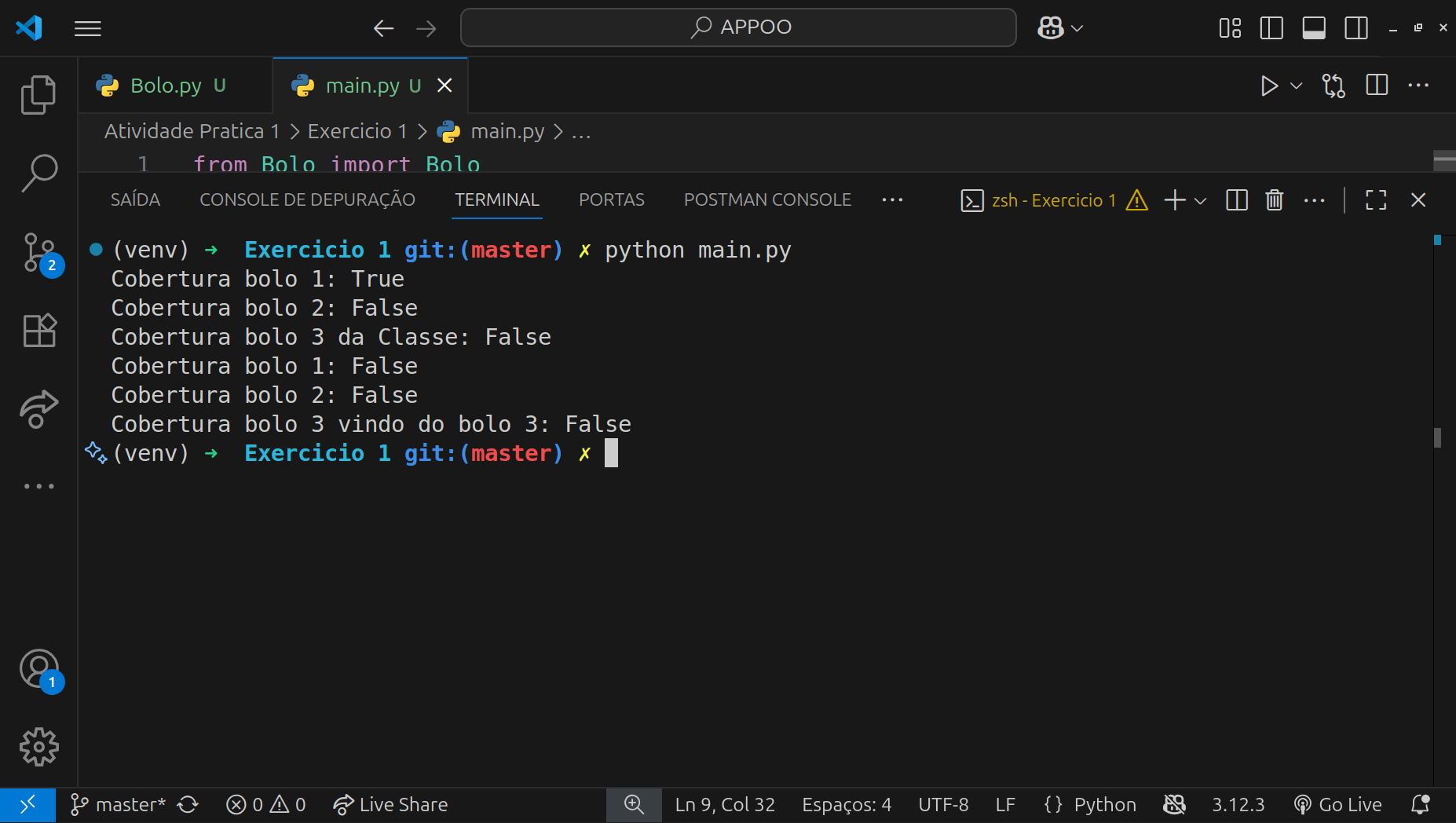
Advindo do primeiro bolo (um objeto já criado), mas mude a cobertura para: sem cobertura

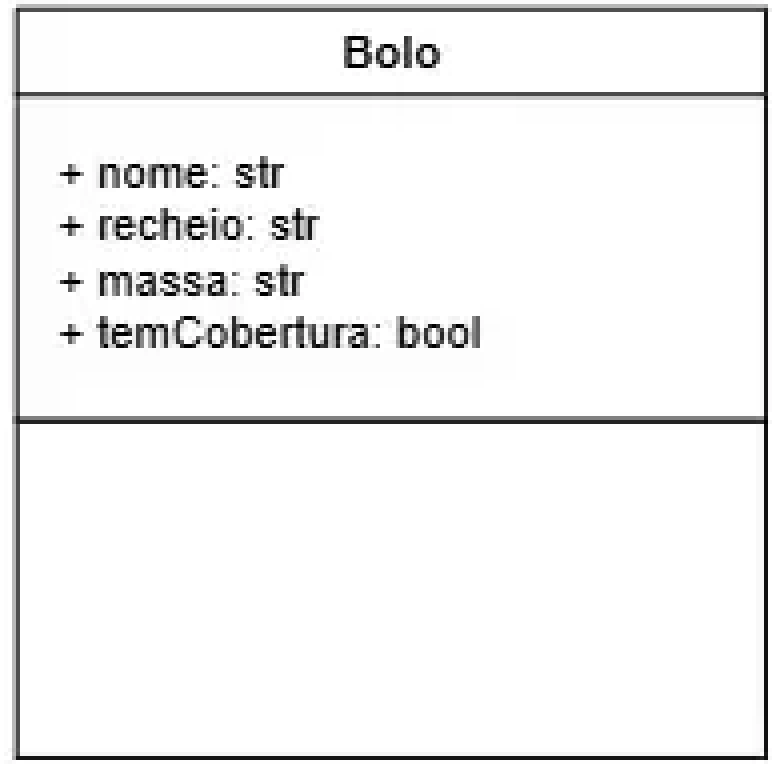
Printe na tela a cobertura de todos os bolos

Explique os resultados



****

****



O Resultado variou entre a criação do terceiro bolo pela classe e pela instância do bolo 1, pois, ao criar diretamente da classe o objeto gerado é uma instância própria, e ao criar pela instância do bolo 1 o objeto gerado é um ponteiro direcionado ao bolo 1, alterando o bolo 3, é alterado o bolo 1.

**2 (Aula 05)** - Faça o diagrama UML e implemente em Python. Coloque os prints dos seus códigos e dos resultados. Considere cada classe em um arquivo .py. Faça testes recursivos de possíveis erros de valor inseridos pelo usuário.

Crie a classe **FazBolo**

Crie um construtor com *massa*, *recheio*, *cobertura*:

Crie um método chamado assar, cujo atributo seja se usará fogo médio, baixo ou alto (temperatura) e o tempo em que o bolo ficará assando:

Printe: “ Você está querendo assar o bolo de *massa* com recheio de *recheio* e cobertura de *cobertura* no fogo *temperatura* por *X* minutos“, sendo X e temperatura inseridos pelo usuário.

Crie prints com avisos que achar necessários considerando as seguintes condições:

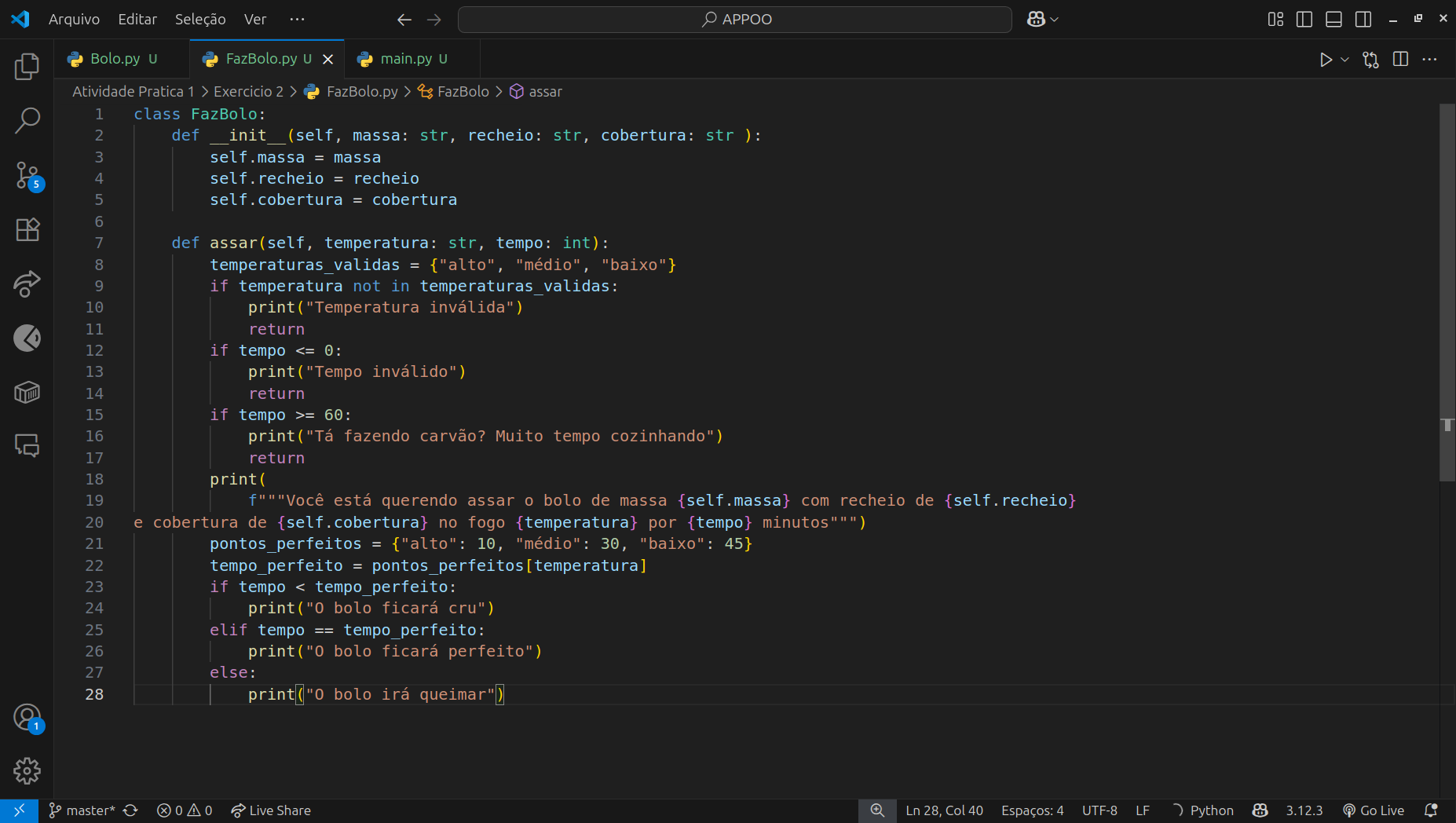
Para o fogo alto, o bolo assa em 10 minutos. Antes fica cru e depois ele queima.

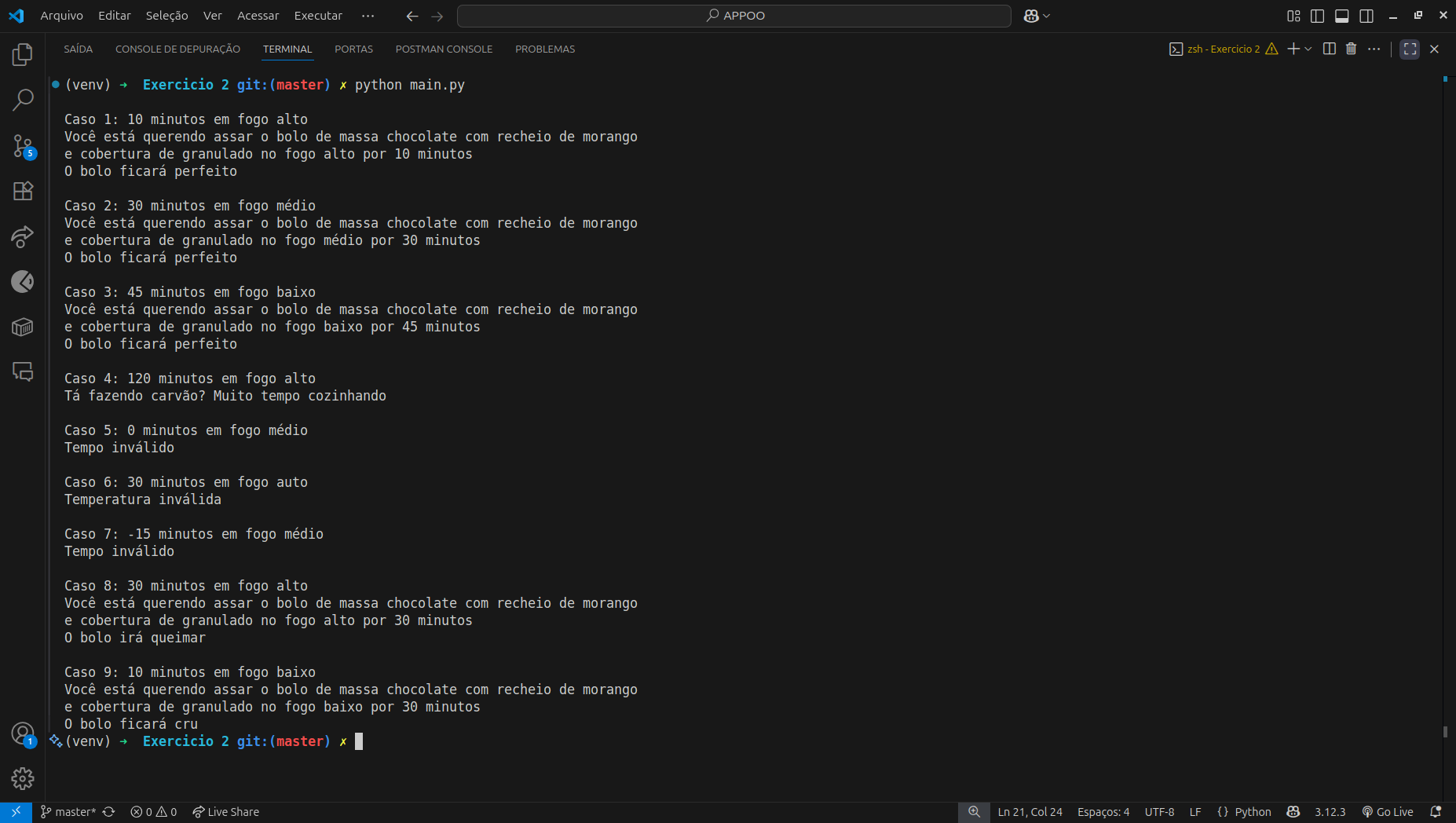
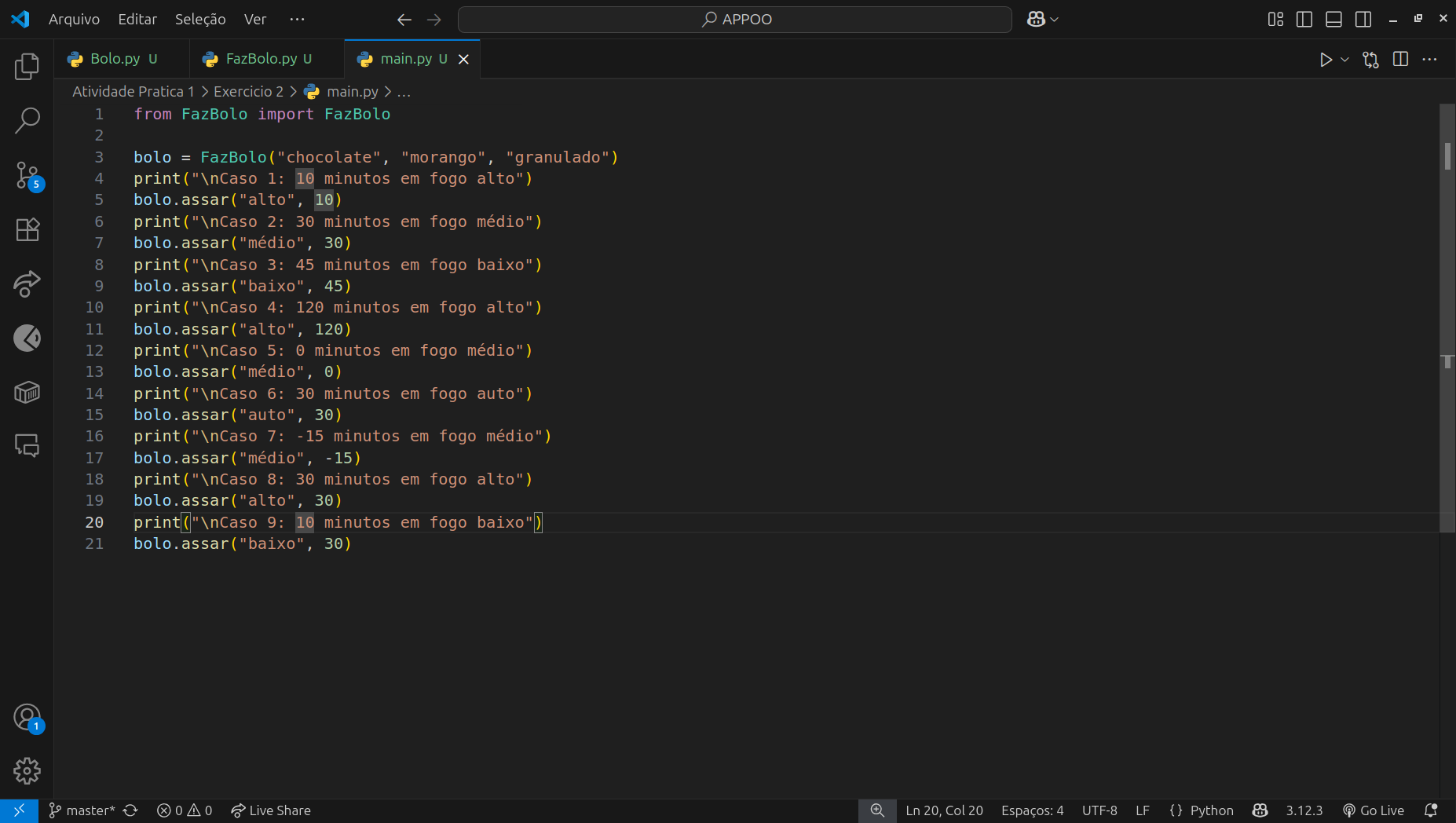
Para o fogo médio, o bolo assa em 30 minutos. Antes fica cru e depois ele queima.

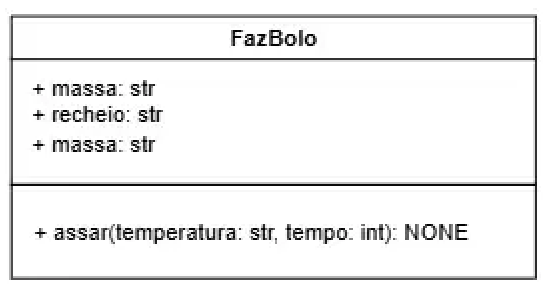
Para o fogo baixo, o bolo assa em 45 minutos. Antes fica cru e depois ele queima.

Considere restrições para números negativos ou valores absurdos!

Crie testes e explique os resultados!



****

****

Com os casos 1, 2 e 3 percebe-se que na temperatura correta o bolo assa corretamente, nos casos 4 a 7 testamos erros, no 8 e 9 casos de cru e assado.

**3 (Aula 05)** - Faça o diagrama UML e implemente em Python. Coloque aqui os prints dos seus códigos e dos resultados dos testes. Crie testes e mensagens de erros para solicitações que geram erro. Considere cada classe em um arquivo .py.

Considere que você está desenvolvendo um sistema para auxiliar na administração de um hotel. Você deve implementar duas classes em Python: Hotel e Cliente. A classe Cliente representa um hóspede do hotel, enquanto a classe Hotel representa o próprio estabelecimento. Siga as instruções abaixo para desenvolver o código:

•Classe Cliente:

•Atributos -> nome, dias\_estadia, consumo\_restaurante;

•Implemente um método chamado fornecaValorConta que calcule e retorne o valor total da conta do cliente.

•Cada dia de estadia tem o custo de R$100,00 e cada refeição no restaurante tem o custo de R$50,00.

•Classe Hotel:

•Crie uma classe chamada Hotel com o atributo nome\_hotel

•Implemente um método chamado determineContaCliente que receba como parâmetro um objeto do tipo Cliente e retorne o valor da conta do cliente.

•Instruções Adicionais:

•Na função main(), crie um cliente e um hotel utilizando dados fictícios.

•Solicite ao hotel que determine a conta do cliente criado.

•Imprima uma mensagem informando o valor da conta do cliente, incluindo todas as informações relevantes: "O cliente [nome do cliente] que ficou [dias de hospedagem] dias no hotel [nome do hotel] teve a conta de [valor da conta] reais, considerando [dias de hospedagem] dias de hospedagem e [refeições] refeições."

**4 (Aula 05)** Inspirado no exercício anterior, crie suas próprias classes para um exemplo que realize mensagem entre objetos. O código deverá ter no mínimo 2 classes e 1 main. Deve haver entrada de dados do usuário e mensagens de saída. Deve haver verificação de erros. O código deve conter um propósito real. Faça o código e o diagrama UML. Mostre prints dos resultados.

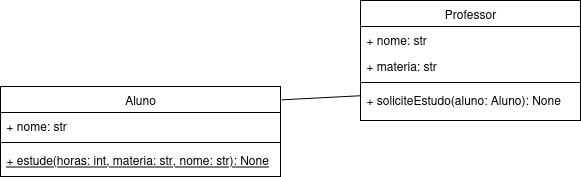
**5 (Aula 07)**- No algoritmo demonstrado abaixo, foi feito o código **de mensagem entre objetos**, em que um professor solicita ao aluno que estude:

Texto

Descrição gerada automaticamente

Visando este código, faça as seguintes modificações:

1. Insira o atributo *horas,* que faça com que o professor requisite ao aluno que estude um determinado número de horas. Além disso, crie um construtor para a classe professor que indique o nome e a matéria que o professor leciona. Com isso, faça com que o método *estude()* imprima: “ O aluno X está estudando para a prova da disciplina Y por W horas, pois o professor Z o requisitou”.
2. Faça o diagrama UML das classes completo do código com essa nova modificação requerida.
3. Crie objetos das classes P*rofessor* e *Aluno* em um código *Main*, como se quisesse testar seu código. Faça o atributo *horas* ser inserido por input com testes para verificar se é um número válido maior que 0.



**6 – (Aula 07)** Crie seu próprio código com o conceito de herança e faça seu diagrama UML. O código deve conter no mínimo 3 classes, com pelo menos 2 atributos e/ou métodos em cada uma. O uso da herança deve ser justificado, isto é, deve haver necessidade de as subclasses usarem o atributo/método da superclasse. Crie cada classe em um arquivo .py e faça os *imports* corretamente. Faça um arquivo main em que esse projeto seja utilizado o mais próximo da prática possível, com tratamento de erro. O código deve ser o mais completo e profissional possível.

**7 – (Aula 09)** Crie um diagrama UML e código Python de formas geométricas para calcular suas áreas e perímetros (Pode ser usado o exemplo 4 como base para esse exercício).

Crie uma classe abstrata chamada FormaGeometrica que represente uma forma geométrica. Esta classe deve conter os seguintes métodos abstratos:

* calcular\_area(): Método abstrato para calcular a área da forma.
* calcular\_perimetro(): Método abstrato para calcular o perímetro da forma.

Além disso, a superclasse deve conter dois métodos comuns:

* Le\_lados(): Lê o valor dos lados que o usuário preencher. Dependendo do numero de lados, ele faz o loop pedindo pro usuário indicar os lados
* Mostra\_lados(): Dependendo do número de lados, ele printa os lados que foram indicados previamente.

Crie três subclasses:

* Retangulo, Quadrado e Triângulo.

Implemente os métodos calcular\_area() e calcular\_perimetro() de acordo com as fórmulas específicas de cada forma. Obtenha os lados pela superclasse FormaGeometrica.

Realize os cálculos, dados recebidos pelos usuários e verificação de erros pelas classes (e não pelo Main). Certifique-se que o retângulo receberá dois lados iguais e que o quadrado terá todos os lados com tamanho igual. No triângulo, implemente a função de verificar se é triângulo retângulo. O código deverá ser inteligente, aproveitando o máximo de funções genéricas na superclasse e evitar verificações específicas das subclasses quando elas não forem fundamentalmente necessárias.

* Ao fim da aula, informe à professora o nome dos membros e o tema do projeto 1 do grupo.

**Aula 05 – Laboratório de Objetos, Classes e Mensagem entre Objetos**

**Aula 07 – Laboratório de Herança**

**Aula 09 – Laboratório de Classes Abstratas**