PROGRAMAÇÃO ORIENTADA A OBJETOS – EXERCICIOS POO + JAVA – CLASSE/OBJETO/HERANÇA

1 (*Hierarquia* ***Employee***) Na aula de hoje, vimos uma hierarquia de herança em que a classe BasePlusCommissionEmployee é herdada da classe CommissionEmployee. Mas nem todos os tipos de empregados são CommissionEmployees.
Neste exercício, você criará uma superclasse Employee mais geral para *calcular* os atributos e comportamentos na classe CommissionEmployee que são comuns a todos os Employees. Os atributos e comportamentos comuns a todos os Employees são firstName, lastName, socialSecurityNumber, getFirstName, getLastName, getSocialSecurityNumber e uma parte do método toString.
Crie uma nova superclasse Employee que contenha esses métodos e variáveis de instância, além de um construtor.
Então, reescreva a classe CommissionEmployee como uma subclasse de Employee. A classe CommissionEmployee só deve conter os métodos e as variáveis de instância que não são declarados na superclasse Employee. O construtor da classe CommissionEmployee deve chamar o construtor da classe Employee, e o método toString de CommissionEmployee deve invocar o método toString de Employee.
Depois de concluir essas modificações, execute os aplicativos CommissionEmployeeTest e BasePlusCommissionEmployeeTest com essas novas classes para garantir que os aplicativos continuem a exibir os mesmos resultados para um objeto CommissionEmployee e um objeto BasePlusCommissionEmployee, respectivamente.

2 (*Criando uma nova subclasse de* ***Employee***) Outros tipos de Employees podem incluir SalariedEmployees, que recebem um salário semanal fixo; PieceWorkers, que são pagos pelo número de peças que produzem; ou HourlyEmployees, que recebem um valor 50% maior para as horas extras.
Crie uma classe HourlyEmployee, que é herdada da classe Employee (exercício anterior), e tem variáveis de instância hours (um double), que representa as horas trabalhadas, e wage (um double), que representa os salários por hora, além de um construtor que recebe como argumentos primeiro nome, sobrenome, número de seguro social, salário por hora e número de horas trabalhadas, métodos *set* e *get* para manipular hours e wage, um método earnings para calcular os rendimentos de um HourlyEmployee com base nas horas trabalhadas e um método toString que retorna a representação String de HourlyEmployee. O método setWage deve assegurar que wage não seja negativo, e setHours, que o valor das horas esteja entre 0 e 168 (o número total de horas em uma semana).
Use a classe HourlyEmployee em um programa de teste.