1. Para efetuar o recolhimento do Imposto de Renda a Receita Federal tem o NOME, CPF, UF (RS, PR e SC) e RENDA ANUAL de cada contribuinte, durante o ano.

EX.: Nome: João da Silva CPF: 123.456.789-00 UF: PR RendaAnual: R$40.000

Para o cálculo do imposto a pagar de cada contribuinte, considere o seguinte:

Nível de Renda Anual	Alíquota
0 a 4.000	0%
4.001 a 9.000	5,8%
9.001 a 25.000	15%
25.001 a 35.000	27,5%
acima de 35.000	30%

Sendo assim, deve-se calcular o imposto a pagar do seguinte modo:

Imposto a pagar = Renda Anual * Alíquota

Faça um programa Java orientado a objetos que leia as informações a serem digitadas pela Receita e indique o imposto a pagar de cada contribuinte digitado. A qualquer instante o usuário pode consultar quais contribuintes têm imposto a pagar acima de um determinado valor (informado pelo usuário a cada consulta).

2. Dada uma frase por meio da interface gráfica com o usuário, faça um programa Java que imprima a frase e cada palavra da frase juntamente com o número de caracteres da palavra. Cada palavra da frase é separada por no mínimo um espaço em branco. Para apresentar o resultado, utilize o componente JTextArea.

Ex.: ISTO É UM TESTE

ISTO	4
É	1
UM	2
TESTE	5

3. Há um rumor que circula na internet informando que George Lucas (o criador da série Star Wars) utiliza uma fórmula para criar os nomes para os personagens em suas histórias (Jar Jar Binks, ObiWan Kenobi etc). A fórmula, supostamente, é:

Seu primeiro nome na série Star Wars:

i. Pegue as três primeiras letras de seu sobrenome
ii. Adicione a ele as duas primeiras letras de seu nome

Seu sobrenome na série Star Wars:

i. Pegue as duas primeiras letras do sobrenome de solteira de sua mãe
ii. Adicione a ele as três primeiras letras do nome da cidade onde você nasceu

E agora sua missão: crie uma classe chamada NameGenerator. Esta classe deve ter um método chamado generateStarWarsName que gera um nome completo Star Wars conforme explicado e cuja assinatura é public static String generateStarWarsName(String[] entrada).

O retorno é a String gerada e o parâmetro de entrada é um vetor de String contendo na primeira posição seu nome completo, na segunda posição o nome de solteira de sua mãe e na terceira posição o nome da cidade onde você nasceu. Investigue a classe String para auxiliá-lo.

Exemplo:

entrada[0] = “João Aparecido da Silva” saída gerada → Siljo Doblu
entrada[1] = “Dores”
entrada[2] = “Blumenau”

4. Faça uma classe Java que tenha como atributo uma String para conter uma expressão aritmética. Esta classe deve chamar-se Expressao e possuir métodos para:
   - instanciar o objeto, recebendo a expressão como parâmetro de inicialização: Expressao(String exp)
   - verificar se a expressão aritmética está correta sintaticamente em relação ao número de parênteses utilizados – todos que estão abrindo estão fechando e vice-versa, sem fechar antes de abrir: public boolean estaCorretaSintaticamente()
   - retornar a quantidade de operadores de divisão utilizados: public int getQtdeDivisores()
   - retornar a posição do primeiro operador aritmético da expressão (+, -, *, /). Caso não exista, retornar -1 : public int getPosicaoOperador()

Exemplo: (a+10)/(23*(10.5-b)-2.59/(b*a))
Expressao exp = new Expressao("(a+10)/(23*(10.5-b)-2.59/(b*a))");
exp.estaCorretaSintaticamente(); → retorna true
exp.getQtdeDivisores(); → retorna 2
exp.getPosicaoOperador() → retorna 2

5. A vampire number has an even number of digits and is formed by multiplying a pair of numbers containing half the number of digits of the result. The digits are taken from the original number in any order. Pairs of trailing zeroes are not allowed. Examples include:

1260 = 21 * 60
1827 = 21 * 87
2187 = 27 * 81

Write a program that finds all the 4-digit vampire numbers.
Vampire numbers first appeared in a 1994 post by Clifford A. Pickover to the Usenet group sci.math, and the article he later wrote was published in chapter 30 of his book *Keys to Infinity*.

6. Crie uma classe chamada *VetorDeReais*, que internamente possui um vetor de n reais (*double*). n é passado como parâmetro na criação do objeto. Implemente métodos para:
   a) Receber os n valores;
   b) Retornar a quantidade de números pares (considerando a parte inteira dos números);
   c) Retornar um novo objeto de *VetorDeReais*, criado a partir da divisão deste objeto (A) por outro objeto recebido como parâmetro (B), sendo novo[i] = A[i] / B[i];
   d) Retornar o valor da multiplicação (M) entre este objeto de *VetorDeReais* (A) e outro objeto recebido como parâmetro (B), de tal modo que M = (A[1] * B[n]) + (A[2] * B[n-1]) + ...
   e) Inverter a posição dos elementos do vetor, ou seja, o primeiro elemento vai para a posição do último elemento e o último elemento vai para a posição do primeiro elemento, o segundo elemento vai para a posição do penúltimo elemento e o penúltimo elemento vai para a posição do segundo elemento, e assim por diante;
   f) Retornar a maior diferença entre dois elementos consecutivos do vetor.

A interface ao usuário deverá permitir a entrada dos valores e possuir botões para acionar os métodos implementados. Para os itens c) e d), que necessitam de outro objeto de VetorDeReais, crie um botão que armazena o objeto manipulado naquele momento, para ser usado quando o usuário desejar fazer uma multiplicação ou divisão.

7. Pegar 5 palavras (de no mínimo 3 letras e no máximo 10), em um tabuleiro 15x15 e montar uma palavra cruzada. Crie uma janela com cinco caixas de texto (para digitar cada uma das palavras) e um botão (para distribuí-las no tabuleiro). A palavra deve ser desmontada em letras, para em seguida distribuir sequencialmente essas letras no tabuleiro. Se a palavra é horizontal ou vertical, quem determina é o seu algoritmo. Porém, as cinco palavras devem se cruzar, ou seja, não pode existir uma palavra isolada.

8. Você está trabalhando em um projeto de automação de uma estação ferroviária. Uma das classes identificadas é PassageirosHora, que foi definida como sendo uma matriz tridimensional de inteiros que em cada posição guarda a quantidade de passageiros transportados por hora, de determinado dia, de determinado mês. Assim a matriz é de ordem 12 meses x 30 dias x 24 horas. Crie esta classe, que deve ter as seguintes funcionalidades:

Funcionalidade	Código de teste PassageirosHora ph = new PassageirosHora();
a) adicione mais um passageiro em determinada hora;	ph.adiciona(1,1,13); // 13 h de 1º de Janeiro
b) retorne a quantidade de passageiros transportados em determinado dia;	int qtde = ph.quantosPassageiros(25,12); //Natal
c) retorne o mês em que houve o menor fluxo de passageiros.	int mês = ph.mesMenorFluxo();
d) retorne o dia, mês e hora em que houve a maior quantidade transportada de passageiros;	int[ ] momento = ph.picoTransporte();

9. Implemente o seguinte plano de testes usando JUnit para a classe Contribuinte (exercício 1):

Casos de Teste 01 a 04 – Verificar se a classe Contribuinte calcula o imposto a pagar corretamente para as seguintes situações:		
	Entrada	**Saída** (tolerância a partir da terceira casa decimal)
	Renda = 3.000,00	Imposto calculado = 0,00
	Renda = 9.000,00	Imposto calculado = 522,00
	Renda = 10.000,00	Imposto calculado = 1.500,00
	Renda = 34.911,73	Imposto calculado = 9.600,72
Caso de Teste 05 – Verificar se a classe Contribuinte rejeita renda negativa		
	Renda = -818,50	Renda mantém-se com o valor anterior.
Casos de Teste 06 a 10 – Verificar se a classe Contribuinte aceita os três estados definidos (RS, SC, PR) e rejeita outros		
	Unid Federação = SC	Unid Federação = SC
	Unid Federação = PR	Unid Federação = PR
	Unid Federação = RS	Unid Federação = RS
	Unid Federação = SP	Unid Federação = não alterado
	Unid Federação = RJ	Unid Federação = não alterado

10. Implemente o seguinte plano de testes usando JUnit para a classe InteiroPositivo (exercício 5-Lista 1):

Casos de Teste 1 e 2 - Verificar se a classe InteiroPositivo calcula o fatorial corretamente para as situações:		
	Valor = 10	Fatorial = 3.628.800
	Valor = 20	Fatorial = 2.432.902.008.176.640.000
Casos de Teste 3 e 4 - Verificar se a classe InteiroPositivo identifica os divisores inteiros para as situações:		
	Valor = 14	String de resultado = “Os divisores inteiros são 1, 2, 7, 14 e a quantidade de divisores é 4”
	Valor = 18	String de resultado = “Os divisores inteiros são 1, 2, 3, 6, 9, 18 e a quantidade de divisores é 6”
Casos de Teste 5 e 6 - Verificar se a classe InteiroPositivo retorna corretamente o vetor com elementos de Fibonacci para as situações:		
	Valor = 9	Vetor = { 1, 1, 2, 3, 5, 8, 13, 21, 34}
	Valor = 15	Vetor = { 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610}

11. Implemente o seguinte plano de testes usando JUnit para a classe VetorDeReais (exercício 6):

Caso de Teste 1 – Verificar se a classe VetorDeReais rejeita a divisão de vetores por serem de tamanhos diferentes:		
	Vetor do objeto 1 = {2 , -1, 3.5} Vetor do objeto 2 = {3}	divide = *null*
Caso de Teste 2 – Verificar se a classe VetorDeReais realiza a divisão dos vetores:		
	Vetor do objeto 1 = {2 , -1, 3.5} Vetor do objeto 2 = {3, 2, 1}	divide = {0.666666 , -0.5 , 3.5 }