```java
import java.util.Scanner;

/* @Autores: Alex, Jheniffer, Victor e Winder.
 * GTI 1 - Senac - Projeto Integrador 01/2017*/

/*Objetivo implementar na linguagem Java o programa que a partir de um número IP e do prefixo da rede fornecidos pelo usuário,
  verifique a qual classe este pertence, caso pertença a classe C, calcule e escreva:
    • a máscara de rede;
    • o endereço do Broadcast;
    • criar uma tabela de sub-redes e intervalo de hosts válidos*/

public class ProjetoIntegrador {

    public static void main(String[] args) {
        // Variáveis
        Scanner leia = new Scanner (System.in);
        String IP, Barra, BinRede, BinBroadcast; // Variável "IP": recebe o endereço IP em formato de texto. Variável "Barra": Recebe parte do código binário do octeto do endereço IP. Variável "BinRede": recebe o código binário da rede. Variável "BinBroadcast": recebe o código binário do broadcast.
        String[] BitRede = {"00000000","0000000","000000","00000","0000","000","00","0",""}; //Vetor com um arranjo de 7 posições de dados pré-definidos para calcular o endereço de rede.
        String[] BitBroadcast = {"11111111","1111111","111111","11111","1111","111","11","1",""}; //Vetor com um arranjo de 7 posições de dados pré-definidos para calcular o endereço de broadcast.
        String[] StrOctetos, BinOctetos = new String[4], BinOcts8c = new String[4]; // Variável "StrOctetos": recebe as quatro partes endereço IP em formato de texto. Variável "BinOctetos": recebe o código binário de cada octeto do endereço IP. Variável  "BinOcts8c": Recebe o código binário de cada octeto do endereço IP com 8 casas independente do número.
        String[] Mascara = {"255.0.0.0","255.128.0.0","255.192.0.0","255.224.0.0","255.240.0.0","255.248.0.0","255.252.0.0","255.254.0.0","255.255.0.0",

                            "255.255.128.0","255.255.192.0","255.255.224.0","255.255.240.0","255.255.248.0","255.255.252.0","255.255.254.0","255.255.255.0",

                            "255.255.255.128","255.255.255.192","255.255.255.224","255.255.255.240","255.255.255.248","255.255.255.252"};// Variável do tipo vetor de texto com um array de dados pré-definidos das mascaras de rede.
        int Pref, NumRede, NumBroadcast,Cont=0, Hosts=0, Subredes=0; //Variável "Pref": recebe o prefixo da rede. Variável "NumRede": Recebe a conversão do número binário para um número inteiro do octeto onde é calculado a rede. Variável "NumBroadcast": Recebe a conversão do número binário para um número inteiro do octeto onde é calculado o broadcast. Variável "Cont": Conta a quantidade de sub-redes. Variável "Hosts": Recebe a quantidade de hosts definidos pela mascara. Variável "Sub-redes": recebe a quantidade de sub-redes definidas pela mascara.
        int[] IntOctetos = new int[4];

        System.out.println("((<<--------------Calculo de Sub-redes | Projeto Integrador 01/2017-------------->>))");
        do{
            do{
                System.out.print("  -> Digite o endereço de IP desejado: ");
                IP = leia.next(); // IP lido no formato de texto.
                StrOctetos = IP.split("\\."); // Texto lido é dividido entre as posições do vetor a cada ponto digitado.
                if (StrOctetos.length != 4){
                System.out.println("  IP inválido! Digite Novamente...");
                 }
            }while(StrOctetos.length != 4); // Caso as posições do vetor sejam menor ou maior que 4, o laço entra em ação e repete a operação pedindo ao usuário para digitar novamente.

            for (int i = 0; i < 4; i++) { // Armazena os dados nas 4 posiçoes do vetor.
```

```
39                  IntOctetos[i] = Integer.parseInt(StrOctetos[i]); // O número
                    armazenado no vetor "StrOctetos" em formato de texto é convertido
                    para um número inteiro e armazenado no vetor "IntOctetos".
40              }
41
42              if (IntOctetos[0]<1 || IntOctetos[0]>255 || IntOctetos[1]<0 ||
                IntOctetos[1]>255 || IntOctetos[2]<0 || IntOctetos[2]>255 ||
                IntOctetos[3]<0 || IntOctetos[3]>255){
43                  System.out.println("  IP inválido! Digite Novamente...");
44              }
45          }while(IntOctetos[0]<1 || IntOctetos[0]>255 || IntOctetos[1]<0 ||
            IntOctetos[1]>255 || IntOctetos[2]<0 || IntOctetos[2]>255 || IntOctetos[3]<0
            || IntOctetos[3]>255); // Caso o valor de qualquer octeto seja menor 0 ou
            maior que 255, o laço entra em ação e repete a operação pedindo ao usuário
            para digitar novamente.
46
47          for (int i = 0; i < 4; i++) { // Armazena os dados nas 4 posiçoes dos vetores.
48              BinOctetos[i] = Integer.toBinaryString(IntOctetos[i]); // Cada número
                armazenado no vetor "IntOctetos" é convertido para o binário e
                armazenado no vetor "BinOctetos".
49              BinOcts8c[i] = String.format ("%08d", Integer.parseInt(BinOctetos[i]));
                // Cada valor binário armazenado no vetor "BinOctetos" é transformado em
                binarios sempre com 8 casas númericas e armazenado no vetor "BinOcts8c".
50          }
51
52          do{
53              System.out.print("\n  -> Por favor, digite o prefixo da rede: /");
54              Pref = leia.nextInt();
55              if (Pref<8 || Pref>30) {
56                  System.out.println("  Prefixo invalido! Digite Novamente");
57              }
58              else if(Pref<23 && IntOctetos[0]>191){
59                  System.out.println("  Prefixo invalido! IP Classe C prefixo deve ser
                    maior ou igual à 24");
60              }
61          }while(Pref<8 || Pref>30 || (Pref<8 && IntOctetos[0]>191)); // Impede que o
            usuario digite o prefixo da rede menor que 8 ou maior que 30 e também que um
            IP classe C receba prefixo menor que 23.
62
63
            System.out.println("\n-----------------------------------Resultados:----------
            ------------------------------");
64          System.out.print("Endereço.....................: "+IP);
65          if (IntOctetos[0] >=1 && IntOctetos[0]<=127) { // Definição das classes do
            endereço digitado.
66              System.out.print(" Classe A\n");
67          }
68          else if (IntOctetos[0] >=128 && IntOctetos[0]<=191) {
69              System.out.print(" Classe B\n");
70          }
71          else if (IntOctetos[0] >=192 && IntOctetos[0]<=223) {
72              System.out.print(" Classe C\n");
73          }
74          else if (IntOctetos[0] >=224 && IntOctetos[0]<=239) {
75              System.out.print(" Classe D\n");
76          }
77          else if (IntOctetos[0] >=240 && IntOctetos[0]<=255) {
78              System.out.print(" Classe E\n");
79          }
80
81          if(Pref>=8 && Pref<=15){ // Calculo do 2° Octeto.
82              Barra = BinOcts8c[1].substring(0, Pref-8); // A função ".substring(0,
                n)" separa parte de uma String por caracteres da esquerda para a direita
                onde "n" é o número de caracteres que serão separados, no caso
                especifico o "prefixo-8" define quantos caracteres vão ser separados.
83              BinRede = Barra+BitRede[Pref-8]; // Concatena o texto separado e
                armazenado na variável "Barra" com o texto presente na posição do vetor
                "BitRede" definida pelo prefixo da rede que foi digitado e armazena na
                variável BinRede.
84              BinBroadcast = Barra+BitBroadcast[Pref-8]; // Concatena o texto separado
                e armazenado na variável "Barra" com o texto presente na posição do
                vetor "BitBroadcast" definida pelo prefixo da rede que foi digitado e
                armazena na variável BinBroadcast.
```

```
85              NumRede = Integer.parseInt(BinRede, 2); // Converte o texto da variável
                BinRede para um número decimal inteiro e armazena na variável NumRede.
86              NumBroadcast = Integer.parseInt(BinBroadcast, 2); // Converte o texto da
                variável BinBroadcast para um número decimal inteiro e armazena na
                variável NumBroadcast.
87              System.out.println("Rede.....................:
                "+IntOctetos[0]+"."+NumRede+".0.0"); // Imprime o endereço de rede.
88              System.out.println("Broadcast................:
                "+IntOctetos[0]+"."+NumBroadcast+".255.255"); // Imprime o endereço de
                Broadcast.
89              System.out.println("Mascara..................: "+Mascara[Pref-8]); //
                A máscara é difinida pelo valor definido na posição do vetor "Mascara" a
                posição do mesmo é definido pelo prefixo da rede digitado.
90              Hosts = (int) Math.pow(2, (32-Pref)); // A quantidade de hosts é
                difinida por 2 elevado ao total de bits de um endereço IP menos o
                prefixo digitado.
91              Subredes = (int) Math.pow(2, (24-(32-Pref))); // A quantidade de
                sub-redes é difinida por 2 elevado a quantidade de bits dos 3 ultimos
                octetos menos total de bits de um endereço IP menos o prefixo digitado.
92              System.out.println("Hosts usáveis por sub-rede..: "+(Hosts-2)); //
                Imprime a quantidade de hosts usaveis por subrede.
93              System.out.println("Total de sub-redes.......: "+Subredes); //
                Imprime o total de subredes por octeto.
94
                System.out.println("-------------------------------------------------------
                ------------------------------");
95              System.out.println("\nPara calcular a tabela de Sub-Redes digite um IP
                Classe C");
96          }
97          else if(Pref>=16 && Pref<=23){ // Calculo do 3° Octeto.
98              Barra = BinOcts8c[2].substring(0, Pref-16);
99              BinRede = Barra+BitRede[Pref-16];
100             BinBroadcast = Barra+BitBroadcast[Pref-16];
101             NumRede = Integer.parseInt(BinRede, 2);
102             NumBroadcast = Integer.parseInt(BinBroadcast, 2);
103             System.out.println("Rede.....................:
                "+IntOctetos[0]+"."+IntOctetos[1]+"."+NumRede+".0");
104             System.out.println("Broadcast................:
                "+IntOctetos[0]+"."+IntOctetos[1]+"."+NumBroadcast+".255");
105             System.out.println("Mascara..................: "+Mascara[Pref-8]);
106             Hosts = (int) Math.pow(2, (32-Pref));
107             Subredes = (int) Math.pow(2, (16-(32-Pref)));
108             System.out.println("Hosts usáveis por sub-rede..: "+(Hosts-2));
109             System.out.println("Total de sub-redes.......: "+Subredes);
110
                System.out.println("-------------------------------------------------------
                ------------------------------");
111             System.out.println("\nPara calcular a tabela de Sub-Redes digite um IP
                Classe C");
112         }
113         else if(Pref>=24 && Pref<=30){ // Calculo do 4° Octeto.
114             Barra = BinOcts8c[3].substring(0, Pref-24);
115             BinRede = Barra+BitRede[Pref-24];
116             BinBroadcast = Barra+BitBroadcast[Pref-24];
117             NumRede = Integer.parseInt(BinRede, 2);
118             NumBroadcast = Integer.parseInt(BinBroadcast, 2);
119             System.out.println("Rede.....................:
                "+IntOctetos[0]+"."+IntOctetos[1]+"."+IntOctetos[2]+"."+NumRede);
120             System.out.println("Broadcast................:
                "+IntOctetos[0]+"."+IntOctetos[1]+"."+IntOctetos[2]+"."+NumBroadcast);
121             System.out.println("Mascara..................: "+Mascara[Pref-8]);
122             Hosts = (int) Math.pow(2, (32-Pref));
123             Subredes = (int) Math.pow(2, (8-(32-Pref)));
124             System.out.println("Hosts usáveis por sub-rede..: "+(Hosts-2));
125             System.out.println("Total de sub-redes.......: "+Subredes);
126
                System.out.println("-------------------------------------------------------
                ------------------------------");
127             NumRede = 0;
128             NumBroadcast = Hosts-1;
129             if(IntOctetos[0] >=192 && IntOctetos[0]<=223){ // impede que quando for
                digitado um IP classe D ou E seja calculado a tabela de subredes.
130                 System.out.println("\n---------------------------------Tabela de
```

```
                    Sub-Redes--------------------------------");
131                 while(NumBroadcast<=255){ // Calculo da tabela de sub-redes.
132                     Cont++;
133                     System.out.print(Cont+"ª Sub-rede:
                        "+IntOctetos[0]+"."+IntOctetos[1]+"."+IntOctetos[2]+"."+NumRede+"
                        | Hosts: "); //Imprime o endereço de rede da tabela.
134                     NumRede=NumRede + 1;
135
                        System.out.print(IntOctetos[0]+"."+IntOctetos[1]+"."+IntOctetos[2]
                        +"."+NumRede+" à "); //Imprime o primeiro IP da faixa de
                        endereços validos da tabela.
136                     NumBroadcast = NumBroadcast -1;
137                     System.out.print(NumBroadcast+" | Broadcast: "); //Imprime o
                        ultimo IP da faixa de endereços validos da tabela.
138                     NumBroadcast = NumBroadcast +1;
139
                        System.out.print(IntOctetos[0]+"."+IntOctetos[1]+"."+IntOctetos[2]
                        +"."+NumBroadcast+"\n"); //Imprime o endereço de broadcast da
                        tabela.
140                     NumRede = (int) ((NumRede -1 ) + Hosts); // Acrescenta o próximo
                        endereço de rede.
141                     NumBroadcast = (int) ((NumBroadcast) + Hosts); // Acrescenta o
                        próximo endereço de broadcast.
142
                        System.out.println("----------------------------------------------
                        ---------------------------------------");
143                 }
144             }
145             else{
146                 System.out.println("\nPara calcular a tabela de Sub-Redes digite um
                    IP Classe C");
147             }
148         }
149     }
150 }
```