- O que o código abaixo faz?

```
boolean doidao (char c){
      boolean resp= false;
      int v = (int) c;
      if (v == 65 || v == 69 || v == 73 || v == 79 || v == 85 || v == 97 || v == 101 || v ==105 ||
          v == 111 || v == 117){
            resp = true;
      }
      return resp;
}
```

O código acima é um método que recebe como parâmetro a variável de caractere c, converte esse caractere em número inteiro (de acordo com a tabela ASCII) e logo depois testa se ela é vogal (que também estão em números da tabela ASCII) - se for, retorna true, caso contrário retorna false.

```
char toUpper(char c){
      return (c >= 'a' && c <=  'z') ? ((char) (c – 32)) : c ;
}
```

O código acima testa de o caractere está em letra maiúscula - caso seja false, ele o transforma em letra maiúscula subtraindo 32 da tabela ASCII; caso seja true, ele apenas retorna o caractere;

```
boolean isVogal (char c){
      c = toUpper(c);
      return (c =='A' || c =='E' ||  c =='I' ||  c =='O' || c =='U');
}
```

O código acima transforma o caractere em letra maiúscula e depois testa se ele é igual as vogais, facilitando a comparação pois só é necessário colocá-las em letra maiúscula.

```
boolean isLetra (char c){
      return (c >= 'A' && c <='Z' ||  c >='a' &&  c <='Z');
}
```

O código acima faz a comparação do caractere e testa se ele é uma letra do alfabeto, retornando true se for, e false se não for.

```
boolean isConsoante (char c){
      return (isLetra(c) == true &&  isVogal(c) == false);
}
```

O código acima utiliza a função isLetra(c) para testar se o caractere é uma letra do alfabeto e a função isVogal(c) para testar se é vogal. Ele resulta em true se o primeiro for true e o segundo for falso (se não é vogal, obrigatoriamente é uma consoante).

- Qual é a diferença entre os dois métodos abaixo?

```
int m1(int i){
      return i--;
}


int m2(int i){
      return --i;
}
```

No primeiro método, o i será retornado e **depois** decrementado, ou seja, vai retornar o valor de i.
No segundo método, o i será decrementado **antes** de ser retornado, ou seja, vai retornar o valor i - 1.

- Um aluno desenvolveu o código abaixo, corrija-o:

```
boolean isConsoante(String s, int n){
    boolean resp= true;
    if (n != s.length()){
        if (s.charAt(n)<'0' || s.charAt(n)>'9'){
            if (isVogal(s.charAt(n))  ==  true){
                resp= false;
            } else {
                resp=isConsoante(s, n + 1);
            }
        } else {
            resp=false;
        }
    }
    return resp;
}
```

O código está correto?

Considerando que o código acima testa se todas as letras da String são consoantes, o código está correto porque primeiramente ele testa se o caractere é uma letra, e caso for, ele testa se a letra é uma vogal - resultando em false, obrigatoriamente ela é uma consoante.

- O que o programa abaixo mostra na tela?

```
byte b = 0; short s = 0; int i = 0; long l = 0;

while (true){
    b++;   s++;   i++;   l++;
    System.out.println(b + " " + s + " " + i + " " + l);
}
```

O programa acima mostra na tela um contador infinito (sem condição de parada no while) - ele incrementa 1 em todas as variáveis e depois printa na tela.

- Por que o código abaixo imprime [46 - 11]?

```
int x = 23, y = 23;
x = x << 1;
y = y >> 1;
System.out.println("[" + x + " - " + y + "]");
```

A variável x tem um bit deslocado para a esquerda, ou seja, é colocado o número 0 na última casa a direita - passando de 0001 0111 (23) para 0010 1110 (46);

A variável y tem um bit deslocado para a direita, ou seja, o número 0 é colocado na última casa a esquerda - passando de 0001 0111 (23) para 0000 1011 (11);

- Qual é a sua opinião sobre o código **REAL** abaixo?

```
Unidade recuperarUnidadeComCodigoDeUCI(Unidade unidadeFilha) {
    Unidade retorno = null;

    if (unidadeFilha.getCodUci() != null && !unidadeFilha.getCodUci().isEmpty()) {
        retorno = unidadeFilha;
    } else {
        retorno = unidadeFilha.getUnidadeSuperior();
    }

    while (retorno == null || retorno.getCodUci() == null || retorno.getCodUci().isEmpty()) {
        retorno = retorno.getUnidadeSuperior();
    }

    return retorno;
}
```

Não compreendi muito bem esse código porque não tenho conhecimento sobre o protocolo UCI, porém ele testa se a unidadeFilha está vazia - caso true, ele retorna a sua Unidade Superior e caso false, retorna a própria unidade filha. Além disso, enquanto (while) o retorno for vazio, ele chama a sua unidade superior.