● JKad25D「char 型と文字コード」

char 型変数 a、b、c それぞれに「A」「B」「C」の文字コードを代入し以下の表示処理を作成せよ。

① 各変数の値を文字として表示する。
② さらに文字コードの表示を追加する。
③ ①②の表示の前に各変数の値をインクリメントして表示する。
④ 「A」「B」「C」を「E」「C」「C」、「H」「A」「L」と変更して表示する。

①まで作成した画面

```
A
B
C
```

②まで作成した画面

```
A
B
C
65
66
67
```

③まで作成した画面

```
B
C
D
66
67
68
```

④は各自で確認すること。

● JKad25C「String 型と文字コード」

入力された文字列の先頭から順に「文字」と「文字コード」を表示する処理を作成せよ。なお、文字コードは 16 進数で表示するものとする。以下のコードで文字コードを 16 進数文字列に変換することが可能。

```
Integer.toHexString(文字コード)　　// 文字コード　→　16 進数文字列変換
```

課題完成時の画面

```
英単語を入力してください>ECCComputer
E の文字コード：45
C の文字コード：43
C の文字コード：43
C の文字コード：43
o の文字コード：6f
m の文字コード：6d
p の文字コード：70
u の文字コード：75
t の文字コード：74
e の文字コード：65
r の文字コード：72
```

● JKad25B「文字コード表！」

文字コード 0x00 から 0xff までの文字を表示せよ。なお、横 16 文字×縦 16 行の表形式で表示するものとし、以下の文字コードのときは「.」（ドット）を表示するようにせよ（そのまま表示すると表示がくずれるので）。

・'¥b'（バックスペース）、'¥t'（タブ）、'¥n'（改行）、'¥r'（復帰）

課題完成時の画面（最初の 2 行はコントロールコード↑を含むので表示が異なる場合があります）

```
□ □ □ □ □ □ □ □ . . . □ □ . □ □
□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □
  ! ″ # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [ ¥ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~ □
? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
? ? ¢ £ ? ¥ ? § ¨ ? ? ≪ ¬ ? ? ―
° ± ? ? ´ μ ¶ · ¸ ? ? ≫ ? ? ? ?
? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? × ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ÷ ? ? ? ? ? ? ? ?
```

文字と文字の間にタブを挟むときれいに表示される。

● JKad25A「文字列の比較！」

文字列 str1 と str2 を以下の 3 通りの方法で比較した結果（true：同じ、または false：異なる）を表示せよ。なお、str1 と str2 の宣言は以下の通り。

```
String str1 = new String("ECC Computer");
String str2 = new String("ECC Computer");
```

① 比較演算子（==）で比較
② String 型の equals メソッドで比較
③ 自分でプログラムを作成し、先頭から 1 文字ずつ比較

課題完成時の画面

```
str1 は"ECC Computer"です！
str2 は"ECC Computer"です！
==（比較演算子）で比較しました！      false
eauqls メソッドで比較しました！       true
一文字ずつがんばって比較しました！    true
```

● JKad25S1「アルファベットをカウントせよ！」

入力された英単語に「A」から「Z」がそれぞれ何文字ずつ存在するかカウントして表示する処理を作成せよ。なお、小文字は同じアルファベットの大文字としてカウントし（たとえば「a」は「A」としてカウント）、英大文字・小文字以外の文字は「その他」としてカウントすること。

課題完成時の画面

```
英単語を入力してください>ECCCollegeOfComputer&Multimedia
A：1
B：0
C：4
D：1
E：5
F：1
G：1
H：0
I：2
J：0
K：0
L：3
M：3
N：0
O：3
P：1
Q：0
R：1
S：0
T：2
U：2
V：0
W：0
X：0
Y：0
Z：0
その他：1
```

スペースは入れないこと（単語の区切りとみなされるので）

● JKad25S2「文字列探索！」

文字列１と文字列２を入力し、文字列１の中に文字列２と同じ文字列があるかどうか探索する処理を作成せよ。なお、同じ文字列を見つけた場合、文字列１の該当箇所を[と]で囲んで表示すること。

ヒント１：探索する方法が思いつかないときは検索すること（「文字列探索　アルゴリズム」←検索）
ヒント２：[と]で囲んで表示する箇所はString型のsubstringメソッドの使用OK（←検索）。for文でがんばってもOK

課題完成時の画面

```
文字列1を入力してください>クリスマス
文字列2を入力してください>リス
同じ文字列を見つけました！
ク[リス]マス
```

● JKad25X「公開鍵暗号（RSA暗号）！」（←興味のある人は検索）

公開鍵案法方式を使って、入力された文字列（英大文字のみ）の暗号化と復元を行う処理を作成せよ。

公開鍵暗号方式…公開鍵を使って暗号化したコードを秘密鍵で復元する。

（秘密鍵：p、q、dの３つ）

```
int p = 7;
int q = 5;
int d = 5;
```

（公開鍵：n、eの２つ）

```
int n = p * q;
int e = 5;
```

暗号化の方法：
① 文字数と同じ要素数のint型配列encodeを準備する。
② 文字コード'A'～'Z'を0～25の数値に変換する。
③ ②の数値をe乗する。
④ ③の数値をnで割ったときの余りを配列encodeに格納する（これをすべての文字に対して行う）。

復元の方法：
① encode配列と同じ要素数のchar型配列decodeを準備する。
② encodeに格納されている数値をd乗する。
③ ②の数値をnで割ったときの余りを計算する。
④ ③の数値（0～25になっている）を'A'～'Z'に変換してdecodeに格納する（これをすべての数値に対して行う）。

課題完成時の画面

```
暗号化する文字列（英大文字のみ）を入力してください>ABCDEFGHIJKLMNOPQRSTUVWXYZ
暗号化します！
0 1 32 33 9 10 6 7 8 4 5 16 17 13 14 15 11 12 23 24 20 21 22 18 19 30   ←― encodeの内容を表示
復元します！
ABCDEFGHIJKLMNOPQRSTUVWXYZ                                             ←― decodeの内容を表示
```