Java – Strings

7.01 Person.java

Write a Person class that has 4 attributes, a first name (String **first**), middle initial (char **middleInitial**), last name (String **last**), and age (int **age**).

The following methods from the String class will be useful: substring, *indexOf*, *charAt*, *length*, *startsWith*, *endsWith*, *toUpperCase*, *toLowerCase*, *equals*, *equalsIgnoreCase* and *compareTo*.

The Character class has numerous static methods and *Character.toUpperCase(char c)* and *Character.toLowerCase(char c)* will be helpful. Both return the char data type.

Constructors

Person() - constructs a default Person "Jane M Doe", 27.

Person(String fullName, int age) - initializes all attributes given a full name and age. You may assume the full name will be in "First Middle Last" format. fullName.indexOf(' ') will be useful.

Instance Methods

boolean **affix**(String *str*) – returns true if last name ends in *str* case insensitive, false otherwise.

boolean **prefix**(String *str*) – true if last name starts with *str* case insensitive, false otherwise.

boolean **hasX**() – returns true if the full name has an 'X' or 'x' in it, false otherwise.

char **firstLetter**() – returns the first letter of a person's first name.

char **lastLetter**() – returns the last letter of a person's last name.

int **length**() – returns the number of characters in a name (middleInitial is only 1 character).

boolean **equals**(Object obj) – returns true if all attributes are equal(case insensitive).

```
// must type cast obj to a Person first

Person other = (Person) obj;

// then check if 'this' object has the same properties as the other object
```

void **toUpperCase**() – modifies a person's name to all uppercase letters.

void **toLowerCase**() – modifies a person's name to all lowercase letters.

void **toTitleCase**() – modifies first, middleInitial and last to titlecase("Jane M Doe").

booelan **isSorted**() – returns true if the name is in sorted order (case sensitive), false otherwise. Sorted order is the same as alphabetical order. A B C is sorted, and B A C is not.

Add **setters**(4), **getters**(4) and a **toString**() – they must follow the proper format.

Sample Data from PersonRunner.java

```
// default Person
Person p = new Person();
out.printf("%s\n", p);  // invoke toString
out.printf("%d\n", p.length()); // check the number of characters
out.printf("%c\n", p.firstLetter()); // first letter of name
out.printf("%c\n", p.lastLetter()); // last letter of name
out.printf("equals: %b\n", p.equals(new Person("Jane mary Doe", 27)));
out.printf("equals: %b\n", p.equals(new Person("Jane Rachel Doe", 26)));
out.printf("prefix Van: %b\n", p.prefix("Van"));
out.printf("prefix Mac: %b\n", p.prefix("Mac"));
out.printf("affix in: %b\n", p.affix("in"));
out.printf("affix e: %b\n", p.affix("e"));
// is first, mi, last in alphabetical order
out.printf("sorted name: %b\n", p.isSorted());
out.printf("has x: %b\n", p.hasX());

p.toUpperCase();
out.printf("toUpperCase: %s\n", p); // name in all caps
p.toLowerCase();
out.printf("toLowerCase: %s\n", p); // name in all lowercase letters
p.toTitleCase();
out.printf("toTitleCase: %s\n\n", p);
```

Sample Execution

```
Person{first=Jane, last=Doe, middleInitial=M, age=27}
8
J
e
equals: true
equals: false
prefix Van: false
prefix Mac: false
affix in: false
affix e: true
sorted name: false
has x: false
toUpperCase: Person{first=JANE, last=DOE, middleInitial=M, age=27}
toLowerCase: Person{first=jane, last=doe, middleInitial=m, age=27}
toTitleCase: Person{first=Jane, last=Doe, middleInitial=M, age=27}
```

Sample Data from PersonRunner.java

```
Person test = new Person("Abexel Frank VanHancockson", 55);
out.printf("%s\n", test);
out.printf("%d\n", test.length());
out.printf("%c\n", test.firstLetter());
out.printf("%c\n", test.lastLetter());
out.printf("equals: %b\n", test.equals(new Person("Jane ary Doe", 27)));
out.printf("equals: %b\n", test.equals(new Person("Abe Frank VanHanCockson", 26)));
out.printf("equals: %b\n", test.equals(new Person("Abe Frank VanHancockson", 55)));
out.printf("prefix Van: %b\n", test.prefix("Van"));
out.printf("prefix Mac: %b\n", test.prefix("Mac"));
out.printf("affix in: %b\n", test.affix("in"));
out.printf("affix son: %b\n", test.affix("son"));
out.printf("sorted name: %b\n", test.isSorted());
out.printf("has x: %b\n", test.hasX());

test.toUpperCase();
out.printf("toUpperCase: %s\n", test);
test.toLowerCase();
out.printf("toLowerCase: %s\n", test);
test.toTitleCase();
out.printf("toTitleCase: %s\n\n", test);
```

Sample Execution

```
Person{first=Abexel, last=VanHancockson, middleInitial=F, age=55}
20
A
n
equals: false
equals: false
equals: false
prefix Van: true
prefix Mac: false
affix in: false
affix son: true
sorted name: true
has x: true
toUpperCase: Person{first=ABEXEL, last=VANHANCOCKSON, middleInitial=F, age=55}
toLowerCase: Person{first=abexel, last=vanhancockson, middleInitial=f, age=55}
toTitleCase: Person{first=Abexel, last=Vanhancockson, middleInitial=F, age=55}
```