Tutorial Sheet - Topic 06

Programming Fundamentals 1

Arrays, and Strings

Part 1 – Arrays

1. Declaring arrays

- Write a statement to declare an array of integers.
- Write a statement to create the array so that it can store 10 integers.

2. Accessing array elements

- Assign the value 12 to the first element of the array.
- Assign the value 18 to the third element of the array.
- Print the value stored at index position 2.

3. Traversing integer arrays

Write a method to read in **6 integers** into an integer array and then **print the values out backwards**.

4. Processing integer arrays

Write a method to read in **6 values** into an integer array. Then **add one to each value** in the array. Finally, **print out the resultant values** in the array.

5. Reading double arrays

Write a method to read in **10 'wages'** into a **double array** (i.e. an array where each element is a `double`, for instance `2.33`).
Finally print out these values.

6. Printing only high wages

Write a method similar to above — read in **10 'wages'** into a **double array**.
This time, only print out **any wages over 100**.

7. Processing a wage reduction

Similar to above — read in **10 'wages'** into a **double array**.
Anyone who earns **over 1000** will have a **10% wage reduction**.
Make this reduction, then print out all the final values.

8. Average of wages

Write a method to read in **10 'wages'** into a **double array**.
Calculate and print the **average** of the inputted wages.

9. Challenge Exercise

Write a method to read in **6 'wages'** into a **double array** as above.
Print out the **largest wage** in the array.

10. Challenge Exercise

Write a method to read in **6 'wages'** into a **double array** as above.
Print out the **largest wage** in the array and the **index** this wage is stored at.

11. Traversing integer arrays

1. Write a loop to print every element of the following array:

```
int[] numbers = {4, 7, 9, 2, 6};
```

2. Modify your loop to print only the **even** numbers in the array.
3. Write a loop to calculate the **sum** of all numbers in the array.
4. Add code to count how many numbers are **greater than 5** and print the count.
5. What happens if the array is empty (length 0)? How can this be checked before looping?

12. Fixed array size vs variable array size - part 1

```
int[] numbers = new int[10];
int sum = 0;
for (int i = 0; i < 10; i++) {
    numbers[i] = input.nextInt();
    sum += numbers[i];
}
System.out.println("The sum of the values you typed in is: " + sum);
```

- What does this program do?
- Why is this version better than using only a single variable?

12. Fixed array size vs variable array size - part 2

```
System.out.println("How many numbers do you need?");
int numData = input.nextInt();
int[] numbers = new int[numData];
int sum = 0;
for (int i = 0; i < numData; i++) {
    numbers[i] = input.nextInt();
    sum += numbers[i];
}
System.out.println("The sum of the values you typed in is: " + sum);
```

- How does this program differ from the previous version?
- Why might we want the user to decide the number of inputs?

13. Populated data vs capacity

- Suppose an array has a capacity of 15, but only 12 elements contain data.
- What value should be used when calculating the average?
- Write a loop to print only the 12 populated results.

Part 2 – Array Classes

1. Arrays of different types

- Write code to create:
    - An array of 10 integers.
    - An array of 4 Strings.
    - An array of 4 Person objects.

2. Working with String arrays

```
String[] words = new String[4];
words[1] = "Dog";
words[3] = "Cat";
for (int i = 0; i < words.length; i++) {
    System.out.println(words[i]);
}
```

- What output will this code produce?
- Why do some lines display `null`?

3. Arrays of objects

Each `Person` object has a method `printFirstName()` that prints the person's first name to the console.

```
Person[] friends = new Person[4];
friends[0] = new Person("Joey", "Tribbiani", 25);
friends[1] = new Person("Rachel", "Green", 24);
friends[2] = new Person("Ross", "Geller", 27);
friends[3] = new Person("Monica", "Geller", 26);
for (int i = 0; i < friends.length; i++) {
    friends[i].printFirstName();
}
```

- What is the purpose of the `new` keyword?
- What happens if one element is not assigned an object?

4. Listing all objects

```
public void listFriends() {
    for (int i = 0; i < friends.length; i++) {
        friends[i].printFirstName();
        friends[i].printSecondName();
    }
}
```

- What does this method do?
- How would you modify it to include each person's age?

Part 3 – Strings

1. Characters and Strings

```
char letter = 'A';
String word = "A";
```

- What type of data is stored in each variable?
- Why are single quotes used for one and double quotes for the other?

Code	Valid?	Reason
`char letter = n;`		
`char letter = "n";`		
`String letter = 'n';`		
`String letter = "n";`		

2. Creating Strings

- Write code to create a String variable `greeting` with the value `"Hello World!"`.

- What type of object is `String`?
- How does a String differ from the primitive `char` type?

3. String methods and dot notation

```
String name = "Joe Soap";
```

- Print the number of characters in `name`.
- Convert `name` to lowercase and print it.
- Remove spaces from the start and end of `name`.

4. Practising String methods

```
String greeting = "   Hello World!   ";
```

Write code to:

- Print the length of `greeting`.
- Trim extra spaces.
- Convert to uppercase.
- Replace `"World"` with `"Java"` and print the new value.

5. Reflection

- Why are Strings called objects rather than primitives?
- How can String methods simplify common programming tasks?