Homework 4 Report

Q1: Varargs short form of variable-length arguments. The feature that allows the method to accept number of arguments. This feature has added in JDK 5. A varargs represented by three dots after the data type. There are some points about varargs.

- This parameter must be the ***last parameter*** declared by the method.
- There can be only ***one variable argument*** in a method
- Variable length arguments could be handled with ***overloading*** and using as an ***array argument.***

Answer is Option B (True)

Q2: The varargs syntax described like :

return_type method name (data_type… variableName){ }

Frisbee not a data type and this line have an error. However, we assume the Frisbee as datatype, we get the first element of varargs parameter with **f[0]** in this question.

Answer is Option B (True)

Q3: Non-primitive data types refers the objects. So these data types known as **reference types.** Non-primitive data types are Strings, Arrays, Classes, and Interface etc. In this question, neither lowercase and nor uppercase do not regarded as primitive.

Answer is Option D (True)

Q4: There are two ways of syntax representation for arrays in java. One of them is brackets are next to the data type other one brackets next to the variables. Declarations of primitive start with data type.

int[] number = new int[44] **int number[]** new int[44]

Answer is Option C (True)

Q5: In java, Arrays use the **toString( )** method to convert the instance to object's String representation. To use this method, import the ***java.util.Arrays*** needed. toString method returns the string representation of the array. This is a form of array elements enclosed with square brackets ([ ]). The argument strings… accepts both of the string and array but the string array take only the array definition.

Answer is Option C (True)

Q6: To determine the number of elements in an array (consist of ***int[], double[], String[]***) used the *arrayName.**length*** method. ***Length()*** can be used for String object, Stringbuilder etc. The ***size()*** method is used to get the number of elements in List interface in java.

Answer is Option A **(True)**

Q7: Two-dimensional arrays represented like in following

data_type*[1st dimension][2nd dimension]…[Nth dimension]* ***array_name***

= ***new data_type****[size1][size2]…. size[N]*

According to this definition there is only option C regards.

Answer is Option C **(True)**

Q8: This question we have an array of strings. The days are printed with the **"for"** statement. The result is occurrence of seven lines.

Answer is Option B **(True)**

Q9: When sorting primitives A**rrays.sort** method uses a dual pivot quick sort. However when sorting objects an iterative implementation of Mergesort. When given an array of n elements **binary search** method searching a sorted array by repeatedly dividing the search interval in half. The idea of binary search is use the information that the array is sorted find the value with reduce the time complexity. *BinarySearch()* is used for searching and *sort()* method is used by sorting operations.

Answer is Option B **(True)**

Q10: The natural ordering of string is lexicographic. In other words, alphabetical order. For example sorting "y11" would be sorted before "y2" because 1 is sorted as smaller than 2 according to the alphabetical order. So the output of this question is 1-10-9 . If we want to sort the strings like they are numbers, we need the convert the strings to numeric values.

Answer is Option B **(True)**

Q11: In arrays, first element always start with **zero** index. The last element of array found at **length -1.** Index. Thus **trains[0]** reference the first element **trains[length -1]** reference the last element in non-empty array.

Answer is Option B **(True)**

Q12:

```
String lion[]   = new String[] {"lion"};
```

This is a correct declaration of string type array.

```
String tiger [] = new String[1] {"tiger"};
```

According to the JLS, array creation expressions with initializers can only have empty brackets. This is **illegal** declaration.

```
String bear []  = new String[] { };
```

This is a correct declaration of string type array.

```
String ohMy []  = new String[0] { };
```

According to the JLS, array creation expressions with initializers can only have empty brackets. This is **illegal** declaration. To sum up, two of them are legal declarations.

Answer is Option C **(True)**

Q13:

```
float lion[]  = new float[];
```

This is **illegal** declaration. Because it needs the initialize with some values or adding curly braces

```
float[] tiger   = new float[1];
```

This is a correct declaration of float type array.

```
float[] bear  = new[] float;
```

There is type mismatch error and this is **illegal** new is a keyword about creating objects.

```
float[] ohMy  = new[1] float;
```

There is an **illegal** expression. **New** is a keyword for creating objects. Usage like data-type is not a valid declaration. To conclude, there is only one legal declaration.

Answer is Option B **(True)**

Q14: The idea of binary search is to use the array is sorted information. And program find the value quickly. So the array must be sorted before making this call. Binary search does not work for **unsorted** list. The complexity is O(logn) in normal condition. For unsorted lists just need a search operation starting from the first element. This gives another complexity to the method.

Answer is Option C **(True)**

Q15: Once the array object is created, its **length** cannot change in the program. To expand the array length, create a new array and refer to it. All variables of an array have the **same type**. Type of an array written in declaration. An array may contain duplicate values. An array start with zero index to reach first element.

Answer is Option A (True)

Q16: In the beginning of code there is an array can take 1*2 = 2 value. However, there are four array element declarations. Two of them adding to the array and other ones excluded. The condition starts from the m3.

Answer is Option C (True)

Q17: String array is created. Containing values are sorted. Binary search method found the right value in this program and returns 1.

Answer is Option B (True)

Q18: There is an error at creating the multidimensional array. So r1 prevent the code compiling.

Answer is Option A (True)

Q19: Integer array is an object, lotto[0] and lotto[1] is an reference class then they can regard as an object. Total three of the objects created.

Answer is Option B (True)

Q20: `[][] String alpha;` `[]String beta;`

This two expressions violates the element type that appears at the beginning of the declaration rule. These are **illegal.**

`String[][] gamma;` `String[] delta[];`

In variable declarations, bracket pairs **allowed** on the type and in declarators.

`String epsilon[][];`

This expression is **valid** even not preferred by programmers. The answer is three

Answer is Option B (True)

Q21: In multidimensional arrays, each curly brace pairs represents the rows. There are three rows in question. First row (a,b,c) second row (d) third row (e,f). This arrays also known as array of arrays. An array create a link to the each row.

Answer is Option B (False) Right option **B**

Q22: This calling of the method names array tries to reach the array element. Array's last element have an index of **length-1.** So the program take **ArrayIndexOutOfBounds** exception.

Answer is Option B (False) Right option **D**

Q23: The size method does not work with arrays. The code gives the compile error.

Answer is Option C (True)

Q24: There are three dimensions in bools array. It transferred to the moreBools the same.

Answer is Option C (True)

Q25: In this question there is an empty array of two index and we printed the memory address of an strings array.

Answer is Option C (True)

Q26: The code is take the ArrayIndexOutOfBounds exception. Because the elements indices bigger than the array indices. This exception is get in line r2.

Answer is Option B (True)

Q27: Copy variable is not defined. The original varargs has not a constant value. The code does not compile.

Answer is Option D (True)

Q28: This code create an Object type object obj and attempt the wrong type of array of objects (int array). Then it takes ArrayStoreException. This exception differs from the Null pointer exception.

Answer is Option D (True)

Q29: Binary search looking for an index of the search key, if it is not contained, returns the negative of insertion point then added -1. The insertion point defined as the point at which the key found into the array. There are 3 of os in the array. Method insert the fourth one (index is 3) the negative one. Result is -3.

Answer is Option C (True)

Q30: This code snippet take the first value of the varargs refers. This is Wolfie then prints it.

Answer is Option B (True)

Q31: This code takes the arguments of 1 and 2. Result is 2

Answer is Option C (True)

Q32: When the command run one reference always show the first element of the args array. Binary search method returns 1.

Answer is Option B (True)

Q33: Declaration of multidimensional arrays can written different square brackets. At option D there is an assign of zeros I one element.

Answer is Option D (True)

Q34: Z variable belongs to the third column. This is also the third element of an array. Index of arra starts at zero. So the answer is dimensions[2][2]

Answer is Option D (False) Right option is **C**

Q35: This code compiled successfully. However, the code is take the ArrayIndexOutOfBounds exception. Because of the interval of the array is shrinked.

Answer is Option D (True)

Q36: The code throws the ArrayIndexOutOfBoundsException because try to reach out of the name array.

Answer is Option C (True)

Q37: This code snippet compiled correctly. There is no prevention occurred.

Answer is Option D **(True)**

Q38: This code gives the error because program looking the size of array used the length keyword.

Answer is Option D **(True)**

Q39: There are two dimensions comes from the boolean, Then there is not any addition about for moreBools array. There are two dimensions takes the moreBools at total.

Answer is Option C **(False)** Right Option is **B**

Q40: There is no argument in this code . Ehen the code compile, console shows only the square braces.

Answer is Option B **(True)**

Q41: The code is compiled. There is no sort operation. Therefore, BinarySearch behaves unstable. The outputs changes time by time. Sometimes it prints 1 and sometimes -1

Answer is Option B **(False)** Right Option **D**

Q42: The code does not compiled X is **string** type but the array is **int** type. There is needs the conversion.

Answer is Option B **(True)**

Q43: There is an multi-dimensional array is consist of first row “Book”, the other row is “Game - 29.99” Listing has two elements and listings first row is has one element. Result is 2 1

Answer is Option A **(True)**

Q44: The code reads the first argument in command line (Firstname) and prints it the console

Answer is Option A **(True)**

Q45: This code compiled. It prints six of the day because for interval is equal to six.

Answer is Option A **(True)**

Q46: Count class takes one argument as an string. (“1 2”) so the length is equal one.

Answer is Option C (False) Right Option **B**

Q47: Binary search method found the Linux word at index zero in this code.

Answer is Option A (True)

Q48: You cannot change method signature from call **(String[] arg)** to call **(String… arg)** because String array only accepts the arrays. Other representation **(String… args)** to call **(String[] args)** is change without issue.

Answer is Option A (True)

Q49: In B option **nums2a** array has four dimensional array and **nums2b** has three dimensional array. Other options they are all consist of four-dimensional arrays.

Answer is Option B (True)

Q50: This code cannot compile because there is a mismatch occurs with args and args[0].

Answer is Option C (True)

The Zen of Python

1. **Beautiful is better than ugly.**
2. **Explicit is better than implicit.**
3. **Simple is better than complex.**
4. **Complex is better than complicated.**
5. **Flat is better than nested.**
6. **Sparse is better than dense.**
7. **Readability counts.**
8. **Special cases aren't special enough to break the rules.**
9. **Although practicality beats purity.**
10. **Errors should never pass silently.**
11. **Unless explicitly silenced**
12. **In the face of ambiguity, refuse the temptation to guess.**
13. **There should be one—and preferably only one –obvious way to do it.**
14. **Although that way may not be obvious at first unless you're Dutch.**
15. **Now is better than never.**
16. **Although never is often better than *right* now.**
17. **If the implementation is hard to explain, it's a bad idea.**
18. **If the implementation is easy to explain, it may be a good idea.**
19. **Namespaces are one honking great idea—let's do more of those!**