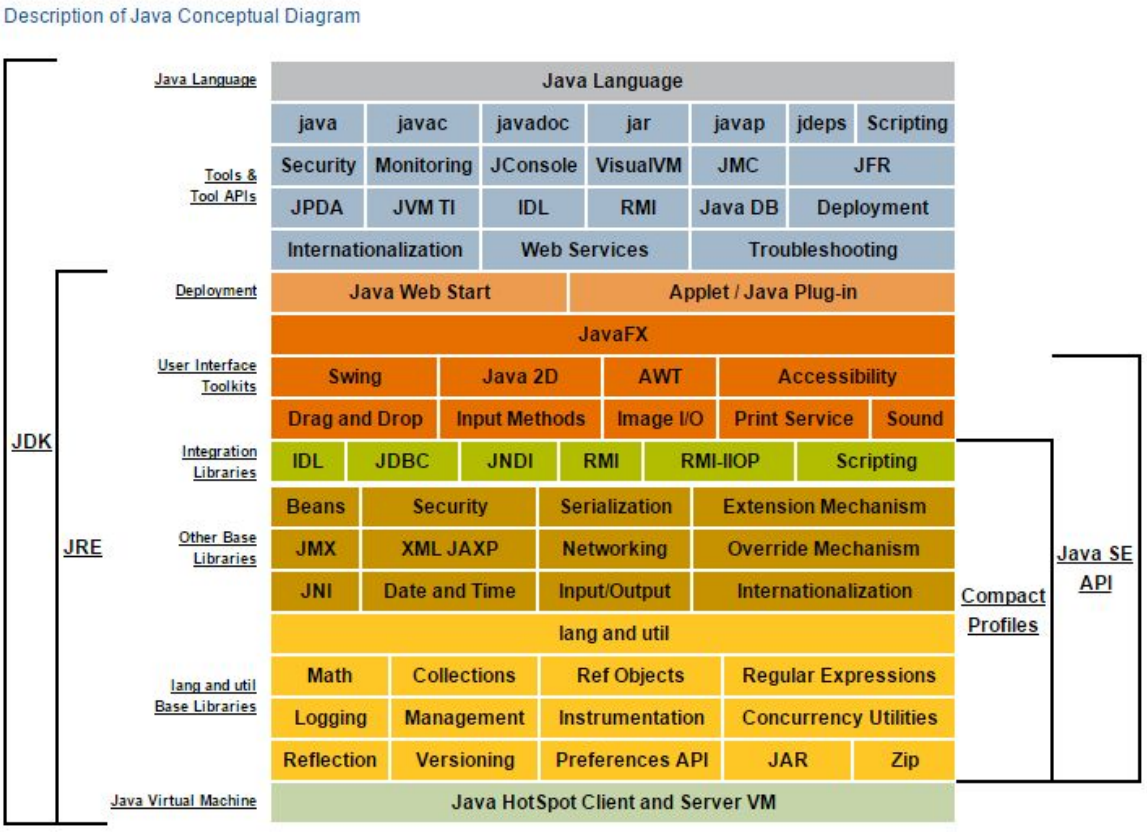
Algorithms & Data Structures

Week 1 Quiz Answers

1 The correct answers are:
JDK and SDK are sometimes referred to interchangeably
JRE is part of the JDK



2 The correct answer is:
Java virtual machine translates Java source code to machine code
Link to Oracle explanation of the JVM:
https://docs.oracle.com/javase/9/vm/JSJVM.pdf

3 The correct answer is:
until - this is not a java keyword

Reserved Words

- The Java reserved words: these are words Java has pre-defined for a special purpose. You cannot create a variable, class name or method name with these words.

abstract	else	interface	super
assert	enum	long	switch
boolean	extends	native	synchronized
break	false	new	this
byte	final	null	throw
case	finally	package	throws
catch	float	private	transient
char	for	protected	true
class	goto	public	try
const	if	return	void
continue	implements	short	volatile
default	import	static	while
do	instanceof	strictfp	
double	int		

4 The correct answer is:
2myvar - incorrectly names variable in java
Void - reserved / keyword in java and not allowed to be used in variable names.

Explanation - The question is about conventions for variable naming in java

Variable Naming

Every programming language has its own set of rules and conventions for the kinds of names that you're allowed to use, and the Java programming language is no different. The rules and conventions for naming your variables can be summarized as follows:

- Variable names are case-sensitive. A variable's name can be any legal identifier — an unlimited-length sequence of Unicode letters and digits, beginning with a letter, the dollar sign "`$`", or the underscore character "_". The convention, however, is to always begin your variable names with a letter, not "`$`" or "_". Additionally, the dollar sign character, by convention, is never used at all. You may find some situations where auto-generated names will contain the dollar sign, but your variable names should always avoid using it. A similar convention exists for the underscore character; while it's technically legal to begin your variable's name with "_", this practice is discouraged. White space is not permitted.
- Subsequent characters may be letters, digits, dollar signs, or underscore characters. Conventions (and common sense) apply to this rule as well. When choosing a name for your variables, use full words instead of cryptic abbreviations. Doing so will make your code easier to read and understand. In many cases it will also make your code self-documenting; fields named `cadence`,

`speed`, and `gear`, for example, are much more intuitive than abbreviated versions, such as `s`, `c`, and `g`. Also keep in mind that the name you choose must not be a keyword or reserved word.

- If the name you choose consists of only one word, spell that word in all lowercase letters. If it consists of more than one word, capitalize the first letter of each subsequent word. The names `gearRatio` and `currentGear` are prime examples of this convention. If your variable stores a constant value, such as `static final int NUM_GEARS = 6`, the convention changes slightly, capitalizing every letter and separating subsequent words with the underscore character. By convention, the underscore character is never used elsewhere. [1]

5 The correct answer is:

Yes - The code is syntactically correct and written according to the rules for creating a method

Example:

Inside main, call the m() method:

```
public class MyClass {
  static void m() {
    System.out.println(5f);
  }
  public static void main(String[] args) {
    m();
  }
}
// Outputs "5"
```

6 The correct answer is:
boolean b =(2=4);
In java boolean can only be true or false

E.g boolean b = true;
boolean b = false;
It's more common to return boolean values from a boolean expression

E.g int x = 10;
Int y = 9;
System.out.println(x > y); // this would return true, because 10 is more than , or higher than, 9

E.g int x = 10;
System.out.println(x == 10); // this would return ture, because we are using the equal to operator == and x = 10 - so the answer is true.

[1] https://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html

7 The correct answer is:
All the answers are correct because none of there are syntactically correct because:

char my-character ='c'; `// incorrect naming of variables  see Q4 information about variable naming`
byte first_variable = 200 `//  you cannot assign 200 to a byte primitive it is out of range. byte range is -128 - 127 inclusive.`
short s = 40000; `// you cannot assign 40000 to a short primitive it is out of range. short range is -32768 - 32767 inclusive.`
boolean b = "true"; `// the boolean is  declared without the quotes around the true as that would make it a string and cause a type error.`

8 The correct answer is 2 2.0 2.5 because:

The results you might expect to get here may well be confusing because there are two different in-built division routines in java, one to calculate an integer answer and the other to calculate the answer as a real number (float, double). Both methods use the single operator / to represent both types of division. This is called overloading the division operator because it can behave in different ways. To decide which type of division to use the compiler looks at the values that are being divided. If at least one value is a real number, it assumes we mean the division routine that calculates an answer as a real number. Otherwise it will use integer division.

The two division modes can be overridden ,or forced, to use one mode or the other by signifying the division method you want to use by casting or by placing an f or a d after one of the values being divided, then it will use the division mode for real numbers as opposed to integer division, but this goes not work for a declared integer primitive. Integer primitives have to be casted as below to become real numbers.

int result1=10/4; `// integer division used here  the real answer is 2.5, but both values are integers so only an integer can be returned, hence the returned value of 2. You can force it to be a real number by casting it like this:` **float d = result1;** `you can't force the cast by adding f or d after one of the values.`

double result2=10/4; `// This gets treated as a real number because it has been declared as a double,which will invoke the real number division, but the values are still integers so the answer will be a whole number followed by a zero. This shows that real number division has been carried out, but they are not real numbers so integer rules apply. Hence the answer is:` **2.0**

float result3 =10f/4; `// Here we have forced the 10 to be a real number, you could do the same to the 4 and get the same result, by adding f after the value, which invokes the real number division method and because we have cast the value to be a real number we get the correct answer 2.5`

`So the correct answer is"`

`2  2.0  2.5`

9 The correct answer is 14 20 1 3.5 **because:**

First of all you need to know the order of precedence for each of the operators in java

Do calculations in parentheses first then follow the operator precedence: Multiply 1, Division 2, Addition 3, Subtraction 4

The first answer is calculated as follows:

System.out.println((27/3 + 2*3)-1 + " "); `// Using the order of precedence rules we calculate the expressions in the inner brackets first, then we calculate the multiplication expression first according to the precedence order. 2*3 = 6 Next we calculate the division expression according to the precedence order. 27/3 = 9 Next, we can carry out the addition between the two expressions in the inner brackets of the two answers. 9 + 6 = 15 The last operation is to subtract 1, which is in the outer brackets.`

`So ((27/3 = 9 + 2 * 3 = 6)= 15 -1) answer = 14 The + after the -1 is the concatenation operator that puts the answer between the “” making it a string that can be printed out and has nothing to do with the calculations.`

The second answer is calculated as follows:

System.out.println(200 % 60 + " ");

`This calculation uses the modulus operator (%) this is a division operator that returns the remainder after revision (it returns an integer - a whole number)`

So how many times does 60 go into 200 - 3 (3 x 60 = 180) with 20 left , which is the remainder, hence the answer is **20**

The third answer is calculated as follows:

System.out.print(6/4+" ");

`This calculation uses integer division as both of the values are integers. Integer division is discussed earlier.`

6/4 = 1.5, which is a decimal (float, double) and integer division only returns integers - whole numbers so the returned answer is **1**

The fourth answer is calculated as follows:

System.out.print(210.00/60+" ");

`The compiler checks the values. One of the values is a real number so it invokes the real number division routine (float, double)`

`210.00 (real number) / 60 (integer) 210.00 / 60 =` **3.5**

10 The correct answer is : 5 4 4 5 because:

The decrement -- and increment operator ++ are used here on the variable x

```
int x=5;
System.out.print(x-- + " ");
// First x is printed  5 then it's decremented by -1 so x is now 4

System.out.print(x  + " ");
// Second x is printed out 4

System.out.print(x++ + " ");
// Third x is printed out 4 then it's incremented by +1 so x is 5

System.out.print(x  + " ");
// Fourth x is printed out 5
```

11 The correct answer is because:

You are looking for the incorrect statement here and the following statement is false.
"The for loop is generally used when the number of repetitions or iterations is unknown in advance"

Because the for loop is used when you know the number of repetition or iterations (loops)