Homework 3 Report

Q1: A **switch** expression is used for have numbered possibilities in workflow. A switch case statement works with **byte short char** (needs casting) and **int** primitive types and **String** (need to convert the case values to string) enumerated types. When I tried with the **double** compiler give the error.
Answer is Option B **(True)**

Q2: Java **ternary** operator also known as conditional operator that takes three operator. This is used instead of **if then else** statement in one line. In general they write as follows:

```
Condition ? True : False
```

In this code snippet there is **pre increment** (++y) and **pre decrement** (- - y) operator. Pre increment operator increase the value **first** and assign operation **later.** Likely pre decrement operator **decrease** the value first and assign the value **later.**

According to question meal value smaller than 6 and second branch actuated. Tip value decreased one and total equals the 6. Question asks about the tip variable. Tip equals 1
Answer is Option D **(False)** Right option **A**

Q3: **Equality operator (= =)** looking of both side for references are equal. **.equals( )** looking for both side equality of the data.

First statement returns false because both "john" values have different address in memory. Other statement returns true because john string has same value with jon object.
Answer is Option C **(True)**

Q4: Plan variable is initialized with 1 then plan++ (post operator increment) variable used with old value (1) and another plan take new value (2) after the evaluation. All plan variables get the same value during evaluating. Other possibilities of operators have been tried. There is syntax error in question at last else line . The else keyword is redundant. Thus, code does not compiled.
Answer is Option B **(False)** Right option **D**

Q5: **Default** keyword can performs when all the cases are **false**. Not all switch statement needs to have a **default** statement. It's optional. Moreover, it can placed before or after any other cases. There must be one case statement to use the default statement. Case statement take a value but **default** statement do not take a value.
Answer is Option C **(True)**

Q6: ThatNumber variable gets 3 from the ternary statement. Then enter the if and incremented to 4. At final exits from the if condition statement.
Answer is Option B **(True)**

Q7: **N**ot every case need to contain a **break** statement. When break statement appears exit from the switch statement without looking remaining cases.
Answer is Option B **(True)**

Q8: Ternary operations should not require **parenthesis,** it's optional. Ternary statements begin with **boolean** expression. When conditional operator is determine the evaluate **one of the** expression not both of them.
Answer is Option C **(True)**

Q9: Logical AND operator (&&) accepts only boolean expressions in java**.** Compiler give an error because Wrapper class type not defined for it. The code does not compile.
Answer is Option C **(True)**

Q10: **%** sign is named **modulus or remainder operator**. This operator takes the remainder of two number after division operation. This question modulus operator get zero because 6 without remainder divide 3.
Answer is Option A **(True)**

Q11: **If-then** statement is not required to have an **else** statement. It depends on reason of the usage. When the if condition statement is false program skip the if else block. No need to cast an obect for if then statement. If-then statement can excute a single statement or a block.
Answer is Option D **(True)**

Q12: In given question when the "if else" statement finish, program flows the next statements. Then flow visits the other if statement, pass it and evaluate the else condition. The result is Not enoughToo many. This option not available in question.
Answer is Option D **(True)**

Q13: A **case** value must become same data type with **switch** variable. If not, program takes compile error (type mismatch). A **case** value include literal statement. A **case** statement can terminated with a break statement. **Break** keyword is optional. The case value must be a constant. The case can be final and need initialized.
Answer is Option A **(False)** Right option **B**

Q14: Given in truth table, the result is true only both of the operators are true. This table tells us it's the logical AND operator (&&). Logical OR operator (||) one the operator is true, expression become true. The decrement and increment operators does not have a truth table.
Answer is Option D **(True)**

Q15: The condition expression of If statement must be a boolean expression for the block of code implemented. The code does not compiled.
Answer is Option C **(True)**

Q16: **T**he **pre-increment** [**++v**] operator increases the value of a variable by one and returns the new value; while the **post-decrement** [**v- -**] operator decreases the value of a variable by one returns the original value.
Answer is Option C **(False)** Right option **B**

Q17: In this code snippet, there is an example of widening primitive conversion. This conversion provides the lose information about the overall magnitude of numeric value. The widening operation can apply any in the following order for each primitive in below. This code compiled and the result is 13.

Byte – short – int – long – float – double
Char – int – long – float – double

Answer is Option B **(True)**

Q18: The variable used in a **switch** statement can **convertible integers** (byte, short, char) **strings** and **enums**. **Long** is the large primitive element and it has issues about convert to **int** or smaller primitive types. **Long** is not permitted by compiler.
Answer is Option B **(True)**

Q19: There is not understandable ternary operator usage. So it can do anything.
Answer is Option D **(True)**

Q20: If there are arithmetic operations alongside, so we can calculate with the operator precedence. Division operation is higher precedence than addition, therefore division operation performed first. In some operations, for example division like arithmetic operation converted the **int** type. Java integer type cannot contain fractions. A syntax error at end of leaders variable declaration.
Answer is Option B **(False)** Right option **C**

Q21: The first element is String then plus (+) operator works as String concatenate operator. If the first element is integer, plus (+) operator works as mathematical operator. Print (5+6+"7"+8+9) is interpreted as 11789.
Answer is Option B **(True)**

Q22: The – operator is used to find the difference between two numbers, while the % operator is used to find the remainder when one number is divided by another.
Answer is Option B **(True)**

Q23: In division operation with integer variables, occurrence of fractions can omitted. At these operations, there is an operator precedence and multiplication is evaluate first.
Answer is Option B **(True)**

Q24: There is no break expression in the switch case statement. Thus, code complied from top to the down consecutively. Eaten value (0) increase one (1), then increase two (3) and decrease one (2).
Answer is Option D **(False)** Right option **B**

Q25: In this question 10 is an integer value and cannot converted to String and type mismatch occurs. Code does not compiled.
Answer is Option C **(True)**

Q26: Given two non-null String objects with reference names apples **(==)** and oranges, if apples oranges evaluates to true, then apples equals() oranges must evaluate to true. Because equality operator **(==)** do some comparison with reference variables are pointing to the same object. Equals() method check that contents are same or not. Not the object itself
Answer is Option A **(True)**

Resource : https://www.java67.com/2012/11/difference-between-operator-and-equals-method-in.html

Q27: If a string value is null and you checked with the xyz.equals(null) method, you will get Null pointer exception (NPE). For any non-null reference value myTestVariable, myTestVariable.equals(null) return false.
Answer is Option B **(True)**

Q28: If statement must includes the boolean values. The code is not compiled.
Answer is Option D **(True)**

Q29: & operator always evaluates both of the expressions. The && operator look at the first expression. It evaluates the second expression only if the first expression is true.
Answer is Option C **(False)** Right option is **B**

Q30: There is not any compile problem. y variable is incremented by 1 and result it 11 5
Answer is Option C **(True)**

Q31: bob String is not null . Bob pointed with notBob and bob String references. Result is true true.
Answer is Option A **(True)**

Q32: At first, the addition operation performed in parenthesis (1+1). Multiplication and Remaining operations are same precedence level. It evaluated left to the right 18 % 2. 18 is a multiple of 2 it returns zero. Finally 12 + 0 equals 12
Answer is Option B **(True)**

Q33: This is exclusive or operand it returns true if and only if the operands have same value. The missing values are true and **false** according to the truth table.
Answer is Option D **(True)**

Q34: There is not any state that code throws exception.
Answer is Option C **(True)**

Q35: The operators + , / , * , % , and ++ are listed in the same or increasing level of operator precedence.
Answer is Option C **(True)**

Q36: ^ operator defines the Bitwise exclusive OR
One operand of ^ true result does not always true . It differs from OR
If both operands of ^ are true the result is not true. It differs from AND
There is no conditional form of operator denoted as ^^.
The ^ operator is boolean exclusive OR and is similar the != operator.
Answer is Option D **(True)**

Q37: Boolean AND operator represents the intersection of two sets.
Boolean OR operator represents the combination fo two sets like in given question.
Answer is Option C **(True)**

Q38: Case expressions must be constant expressions. This code has compiler error. None of the above option is right choice.
Answer is Option D **(True)**

Q39: A number is equal to or greater than 5.21 but strictly less than 8.1 ➔ 8.1 > x >= 5.21
Answer is Option C **(True)**

Q40: The turtle variable has a value of 30, belongs that hare has a value of 25. Turtle is greater than the hare value. Result is Turtle wins!
Answer is Option B **(True)**

Q41: All these arguments is less than 5, all terms are equal zero, addition of zeros become zero.
Answer is Option A **(True)**

Q42: The **spinner** boolean variable assigned by the boolean variable **roller** in if condition statement. The result is up.
Answer is Option A **(True)**

Q43: The logical or || operator is true if either of the of the operands are true, while the logical not **!** operator flips a boolean value.
Answer is Option D **(True)**

Q44: We can think of them like nested if else statements. The code compiled and returns 2.0
Answer is Option A **(True)**

Q45: A switch statement can have any number of **case** statements and at most one **default** statement.
Answer is Option D **(False)** Right option **B**

Q46: The code throw an java.lang.ArrayIndexOutOfBoundsException : arrays length is 0. Because there is not any argument in this code. If we add something into program as arguments “Go outside” and “Stay inside” strings may printed.
Answer is Option B **(False)** Right option **A**

Q47: The code is not working because of ! sign near the 2. 2 is an integer value.
Answer is Option D **(True)**

Q48: The expression w || z means that w and z in logical operator OR. Either of operands are true, || operation is also true. Result is true true.
Answer is Option C **(True)**

Q49: The operators **- , + , / , * ,** and % are listed in the same or increasing level of operator precedence.
Answer is Option A **(True)**

Q50: The code does not compile due to p1. Game variable defined as a String. It is not an integer value.
Answer is Option C **(True)**