Soal Latihan
Branching & Looping

1. Which of the following are keywords or reserved words in Java?
1) if
2) then
3) goto
4) while
5) case
jwb: 1) if 3) goto 4) while 5) case
***then* is not a Java keyword, though if you are from a VB background you might think it was. Goto is a reserved word in Java.**

2. What will be printed out if you attempt to compile and run the following code ?
```
int i=1;
 switch (i) {
 case 0:
 System.out.println("zero");
 break;
 case 1:
 System.out.println("one");
 case 2:
 System.out.println("two");
 default:
 System.out.println("default");
}
```
1) one
2) one, default
3) one, two, default
4) default
jwab: 3) one, two, default
Code will continue to fall through a case statement until it encounters a break.

3. What will be printed out if you attempt to compile and run the following code?
```
int i=9;
switch (i) {
 default:
 System.out.println("default");
 case 0:
 System.out.println("zero");
 break;
 case 1:
 System.out.println("one");
 case 2:
 System.out.println("two");
}
```
1) default
2) default, zero
3) error default clause not defined
4) no output displayed
jwb: 2) default, zero

Although it is normally placed last the default statement does not have to be the last item as you fall through the case block. Because there is no case label found matching the expression the default label is executed and the code continues to fall through until it encounters a break.

4. Which of the following lines of code will compile without error?

1)
```
int i=0;
if(i) {
 System.out.println("Hello");
 }
```
2)
```
boolean b=true;
boolean b2=true;
if(b==b2) {
 System.out.println("So true");
 }
```
3)
```
int i=1;
int j=2;
if(i==1|| j==2)
 System.out.println("OK");
```
4)
```
int i=1;
int j=2;
if(i==1 &| j==2)
 System.out.println("OK");
```
jwab: 2,3
Example 1 will not compile because if must always test a boolean. This can catch out C/C++ programmers who expect the test to be for either 0 or not 0.

5. What will be output by the following code?
```
public class MyFor{
    public static void main(String argv[]){
        int i;
        int j;
    outer:
        for (i=1;i <3;i++)
            inner:
        for(j=1; j<3; j++) {
            if (j==2)
                continue outer;
            System.out.println("Value for i=" + i + " Value for j=" +j);
        }
    }

}
```

1) Value for i=1 Value for j=1
2) Value for i=2 Value for j=1
3) Value for i=2 Value for j=2
4) Value for i=3 Value for j=1

jwab: 1,2
Value for i=1 Value for j=1
Value for i=2 Value for j=1
The
statement continue outer causes the code to jump to the label outer and the for loop increments t

6. What will be the result of executing the following code?

```
1.    boolean a = true;
2.    boolean b = false;
3.    boolean c = true;
4.    if (a == true)
5.    if (b == true)
6.    if (c == true)          System.out.println("Some things are true
in this world");
7.    else                            System.out.println("Nothing is
true in this world!");
8.    else if (a && (b = c))  System.out.println("It's too confusing to
tell what is true and what is false");
9.    else                            System.out.println("Hey this
won't compile");
```

Choices:
a. The code won't compile
b. "Some things are true in this world" will be printed
c. "Hey this won't compile" will be printed
d. None of these

D is correct. This is a very good question to test the concepts of execution flow in case of if conditions. The rule for attaching else statements with if conditions is the same as attaching close brackets with open brackets. A close bracket attaches with the closest open bracket, which is not already closed. Similarly an else statement attaches with the closest if statement, which doesn't have an else statement already, attached to it. So the else statement at line 7 attaches to the if statement at line 6. The else statement at line 8 attaches to the if statement at line 5. The else statement at line 9 attaches to the if statement at line 8. Now let's look at the execution. At line 4 since a is equal to true the execution falls to line 5. At line 5 since b is not true the execution goes to the corresponding else statement at line 8. Now it evaluates the condition inside the if statement. Please note here that an assignment statement also has a value equal to the value being assigned, hence (b = c) evaluates to true and subsequently a && (b = c) e

7. What will be the result of executing the following code?

```
public static void main(String args[])
{
        char digit = 'a';
        for (int i = 0; i < 10; i++)
        {
                switch (digit)
                {
                                case 'x' :
                                {
                                        int j = 0;
                                        System.out.println(j);
                                }
                                default :
                                {
                                        int j = 100;
                                        System.out.println(j);
                                }
                }
        }

        int i = j;
        System.out.println(i);
}
```

Choices:
a. 100 will be printed 11 times.
b. 100 will be printed 10 times and then there will be a runtime exception.
c. The code will not compile because the variable i cannot be declared twice within the main() method.
d. The code will not compile because the variable j cannot be declared twice within the switch statement.
e. None of these.

E is correct. The code will not compile. There is no problem with the declaration of another variable i as both the variables are in disjoint blocks (first one is inside the for loop and its scope ends with the for loop, whereas the second is outside the for loop) and, therefore, different scopes and hence valid. The problem is with the variable j. The two declarations of the variable j are perfectly valid as they are in disjoint blocks and, therefore, different scopes. The error is that both the declarations of j are not available outside the case or default statement, whereas we are trying to assign it to the variable i. Therefore the compiler objects and reports variable j not found.

8. What is the result when you compile and run the following code?

```
public class Test
{

        public void method()
        {
                for(int i = 0; i < 3; i++)
                        {
```

```
                        System.out.print(i);
                    }
                    System.out.print(i);
        }
}
```

Choices:
a. 0122
b. 0123
c. Compilation error
d. None of these

C is correct. The code on compilation will give compile time error because the scope of variable i is only within "for" loop.

9. What does the following expression return?

```
(0.0 == -0.0)
```

1. true
2. false

jawab: 1

10. What will happen when you attempt to compile and run the following code?

```
public class Agg{
static public long i=10;
public static void main(String argv[]){
        switch(i){
                default:
                System.out.println("no value given");
                case 1:
                System.out.println("one");
                case 10:
                System.out.println("ten");
                case 5:
                System.out.println("five");
        }
    }
}
```

1) Compile time error
2) Output of "ten" followed by "five"
3) Output of "ten"
4) Compilation and run time error because of location of default

jwb: 1) Compile time error

This might be considered a "gocha" or deliberate attempt to mislead you because i has been given the data type of long and the parameter must be of long and the parameter must be either a byte, char, short or int. If you attempt to compile this code with JDK 1.2 you will get an error that says something like "Incompatible type for switch, Explicit cast needed to convert long to int". Answering with option 2

11. What will happen when you attempt to compile and run the following code

```
public class MySwitch{
public static void main(String argv[]){
    int k=10;
        switch(k){
        default: //Put the default at the bottom, not here
            System.out.println("This is the default output");
            break;
         case 10:
            System.out.println("ten");
         case 20:
            System.out.println("twenty");
        break;
       }
    }
}
```

1) None of these options
2) Compile time error target of switch must be an integral type
3) Compile and run with output "This is the default output"
4) Compile and run with output of the single line "ten"
jwb: 1) None of these options

Because of the lack of a break statement after the break 10; statement the actual output will be "ten" followed by "twenty"

12. Select one correct statement about the following code.

```
public class Question28 {
        public static void main(String[] args){
                int j=2;
                //line 1
                lab1:for(int i=1;i<j++;i++){
                        //line 2
                        lab2:for(int k=0;k<j--;k++){
                                //line 3
                                if(j>k/i)
```

```
                                        break lab1;
                        }
                }
                System.out.println(j);
        }
}
```

A. Prints: `0`
B. Prints: `1`
C. Prints: `2`
D. Prints: `3`
E. Compilation error.

Jawab: C

This code seems to be overly complicated, but it's not. We have to figure out what the flow is and in such questions, it is always simple, although it doesn't seem at the first sight. A good way to handle this is to write the values of variables i,j and k for each line 1, 2 and 3 in a table .

line # i j k j>k/i
line 1 - 2 - -
line 2 1 3 - -
line 3 1 2 0 true

So at the first iteration, the condition j>k/i yields true, and thus, the outer loop breaks. At that moment, j's final value is 2. The bottom line is don't panic when you are faced with apparently overly complicated nested loops, just try to figure out a simple way out.

13. Select one correct statement about the following code.

```
public class Question38 {
        public static void main(String[] args){
                while(false);           //line 1
                if(false);              //line 2
                do{}while(false);       //line 3
                for(;false;);           //line 4
        }
}
A. Compilation error on lines 1,2 and 4.
B. Compilation error on lines 2,3 and 4.
C. Compilation error on lines 1,2 and 3.
D. Compilation error on lines 1 and 4.
E. The code compiles and runs fine.
jawab: D
```

This is an unreachable statements problem. Just know that the only allowed solutions are the second (so that programmers may define flag variables for debugging,...) and the third (because the body of a do-loop is always at least executed on

14. What is the output of the following code when compiled and run? Select two correct answers.

```
public class Question56 {
	public static void main(String[] args) {
		int i=0;
		while(i++<5){
			switch(i%2){
				default:System.out.print("default
");break;
				case 1:System.out.print("1 ");break;
			}
		}
	}
}
```

A. Prints: 1 default 1 default 1
B. Prints: default 1 default 1 default
C. Prints: 1 1 default 1 default
D. Prints: default default 1 default 1
E. The code compiles and runs fine.

Jwab:AE

The while loop will iterate 5 times (from i=0 to i=4). Then, the expression i%2 may only yield two different values 0 or 1. If the value is 0 (i is even) the default branch is taken otherwise the case 1 branch will be executed. The condition i=0 yields true and then i is incremented (!) to 1. Thus, the first executed branch will be the case and 1 is printed. Then, it will be default's turn and so on.

15. Which of the following statements are infinite loops? Select two correct answers

```
A. for(int i=10;i>0;i-=-2);
B. for(int i=0;i<012;i=i++);
C. for(int i=0;i<0;i--);
D. for(int i=0;(i++^--i)==0;i++);
E. for(int i=010;i==10;i+=0);
```

Jwb:B dan D

A// break|| for(int i=10;i>0;i-=-2);: i is initialized to 10 and then we substract -2 (that is we add 2) to i at each iteration. The loop ends because i's value keeps increasing until it reaches Integer.MAX_VALUE and wraps around to Integer.MIN_VALUE which is smaller than 0.

B//infinite || for(int i=0;i<012;i=i++);: 012 is an octal integer literal whose decimal value is 10. But the trick is in the iteration expression where i is always assigned the value 0. How does that i=i++ work? When executing i++, i's value (0) is stored and then incremented to 1. The value used in the assignment expression is the stored value (0). i will always be 0 and it will never be bigger than10.

C// break || for(int i=0;i<0;i--);: clearly the loop breaks at the first iteration.

D// infinite || for(int i=0;(i++^--i)==0;i++);: the condition expression i++^--i will always yield 0. We have seen that in i++ we use the actual value of i and then we increment it. In --i, we first decrement i and then we use the decremented value. Thus, i++^--i can be rewritten as i^i and this always yields 0. The condition will never be false and the loop never ends.

E// break || for(int i=010;i==10;i+=0);: again, 010 is an octal integer literal whose