Laboratorium 3

CEL: Operatory. Sterowanie kolejnością wykonania.

Polecenia:

Zadanie#1. Przeanalizuj poniższy kod.

```
public void trueOrFalse( )
{
    http://blackbeltfactory.com
    break http;
}
```

Wskaż, czy powyższy fragment kodu zostanie skompilowany poprawnie?

a)	Tak
b)	Nie

■

Zadanie#2. Czy prawdą jest, że do weryfikacji, iż dwie referencje wskazują na ten sam obiekt należy używać metody `equals()`?

a)	Tak
b)	Nie

■

Zadanie#3. Przeanalizuj poniższy kod.

```
class Test {
    public static void main(String[] args) {
        int a[] = {1, 2};

        Integer i = 0;

        for (i : a) {
            i += i;
        }
        System.out.println(i);
    }
}
```

Wskaż, jaki wynik wykonania powyższego kodu zostanie wyświetlony na konsoli?

a)	2
b)	3
c)	4
d)	Wystąpi błąd kompilacji

■

Zadanie#4. Przeanalizuj poniższy fragment kodu.

```
int i = 3, j = 0, result = 1;
result += i-- * --j ;
System.out.println( result );
```

Wskaż, jaki wynik wykonania powyższego kodu zostanie wyświetlony na konsoli?

a)	0
b)	W trakcie wykonania zostanie wyrzucony wyjątek
c)	Błąd kompilacji
d)	-3
e)	-2
f)	-1

■

Zadanie#5. Przeanalizuj poniższy fragment kodu.

```
int a = 3;
int b = 2;
System.out.println(a/b);
```

Wskaż, jaki wynik wykonania powyższego kodu zostanie wyświetlony na konsoli?

a)	1
b)	1.5

■

Zadanie#6. Przeanalizuj poniższy kod.

```
int[]a = {1, 2, 3, 4, 5};
int suma = 0;

for( int i = 0; i < a.length; i++ )
{
    suma += a[ i ];
}
```

W powyższym kodzie zamień pętlę `for` na pętlę `for-each`.

■

Zadanie#7. Przeanalizuj poniższy fragment kodu.

```
01: public class Example {
02:   public static void main(String[] args) {
03:      int nOp = 5;
04:      if(nOp%2 < 1) {
05:          System.out.print("Even");
06:      }
```

```
07:       else {
08:           System.out.print("Odd");
09:       }
10:     }
11: }
```

Wskaż, jaki wynik wykonania powyższego kodu zostanie wyświetlony na konsoli?

a)	Nic
b)	Wystąpi błąd kompilacji
c)	`Even`
d)	`Odd`
e)	`EvenOdd`
f)	`Even` `Odd`

■

Zadanie#8. Przeanalizuj poniższy kod.

```
System.out.println(2.00 - 1.10);
```

Wskaż, jaki wynik wykonania powyższego kodu zostanie wyświetlony na konsoli?

a)	`0.8999999999999999`
b)	`0.9`

■

Zadanie#9. Przeanalizuj poniższy fragment kodu.

```
01: public static void main(String[] args) {
02:     {
03:         int x = 2;
04:         x = +2;
05:         int y = 10;
06:         x += y;
07:     }
08:     System.out.println(x);
09: }
```

Wskaż, jaki wynik wykonania powyższego kodu zostanie wyświetlony na konsoli?

a)	`14`
b)	`12`
c)	Wystąpi błąd kompilacji

■

Zadanie#10. Przeanalizuj poniższy fragment kodu.

```
01: boolean correct = false;
02: boolean incr = true;
03: int i = correct ? 15 : (incr ? 16 : 14.0);
```

Wskaż, jaka wartość zostanie przypisana do zmiennej `i`?

a)	`14`
b)	`15`
c)	`16`
d)	Wystąpi błąd kompilacji

■

Zadanie#11. Przeanalizuj poniższy fragment kodu.

```
01: public class PrimitiveTypes {
02:     public static void main(String... args){
03:         int x = 127;
04:         byte b = (byte) x;
05:         System.out.println(b);
06:     }
07: }
```

Wskaż, jaki wynik wykonania powyższego kodu zostanie wyświetlony na konsoli?

a)	`true`
b)	`127`
c)	Wystąpi błąd kompilacji

■

Zadanie#12. Przeanalizuj poniższy fragment kodu.

```
01: int x = 10;
02: int y = 5;
03: if ( x > 0 || ++y < 10 ) {
04:     System.out.print(( x + y) + " ");
05: }
06: System.out.print("finish");
```

Wskaż, jaki wynik wykonania powyższego kodu zostanie wyświetlony na konsoli?

a)	`1005 finish`
b)	`1006 finish`
c)	`15 finish`
d)	`16 finish`
e)	Wystąpi błąd

■

Zadanie#13. Przeanalizuj poniższy fragment kodu.

```
01: public class TrickySwitch {
02:   enum Bound {LEFT, RIGHT, DOWN, UP};
03:
04:   public static void main(String[] args) {
05:     Bound []bounds = {Bound.LEFT, Bound.RIGHT, Bound.DOWN, Bound.UP};
```

```
06:     for (Bound bound : bounds) {
07:       switch(bound) {
08:         case LEFT:
09:              DOWN:
10:                System.out.println(bound);
11:                break;
12:         case RIGHT:
13:              UP:
14:                System.out.println(bound);
15:                break;
16:       }
17:     }
18:   }
19: }
```

Wskaż, jaki wynik wykonania powyższego kodu zostanie wyświetlony na konsoli?

a)	LEFT RIGHT
b)	LEFT RIGHT DOWN UP
c)	Wystąpi błąd kompilacji
d)	Żadne z powyższych

■

Zadanie#14. Przeanalizuj poniższy fragment kodu.

```
01: public class StringTest {
02:     public static void main(String[] args) {
03:         String s1 = "Anthony";
04:         String s2 = "Anthony";
05:         String s3 = new String("Muller");
06:         String s4 = new String("Muller");
07:
08:         if (s1 == s2) {
09:             System.out.print("Anthony");
10:         }
11:
12:         if (s3 == s4) {
13:             System.out.print("Muller");
14:         }
15:     }
16: }
```

Wskaż, jaki wynik wykonania powyższego kodu zostanie wyświetlony na konsoli?

a)	`Anthony`
b)	`AnthonyMuller`
c)	`Muller`
d)	Niemożliwe do przewidzenia
e)	Żadne z powyższych

■

Zadanie#15. Przeanalizuj poniższy kod.

```
byte b1 = 5;
byte b2 = 10;
x = b1 * b2;
```

Jakiego typu może być zmienna `x` (jedna odpowiedź)?

a)	(A) (B) (D) (E)
b)	(B) (C) (D)
c)	(B) (D) (E) (F)
d)	(B) (D) (E)
e)	(D) (F)

gdzie:

```
(A) byte
(B) int
(C) short
(D) long
(E) float
(F) double
```

■

Zadanie#16. Czy prawdą jest, że gałąź `default` musi być ostatnią gałęzią instrukcji `switch`?

a)	Tak
b)	Nie

■

Zadanie#17. Czy prawdą jest, że w instrukcji `switch` niepusta (zawierająca instrukcje) gałąź `case` poprzedzająca kolejną gałąź `case`, która jednocześnie nie jest ostatnią gałęzią tej instrukcji, musi zostać zakończona instrukcją `break`?

a)	Tak
b)	Nie

■

Zadanie#18. Proszę napisać fragment kodu Java sumujący liczby całkowite nieparzyste z przedziału <1,99> przy użyciu instrukcji `for` i operatora `%`. Proszę założyć, że zmienne `sum` oraz `count` zostały już zadeklarowane.

■

Zadanie#19. Proszę podać nazwę instrukcji sterowania, której ciało (*body*) zostanie wykonane co najmniej raz bez względu na wartość wyrażenia warunkującego kontynuację powtarzania pętli.

■

Zadanie#20. Poniższy fragment kodu proszę zmodyfikować w taki sposób, aby osiągnięcie przez zmienną `j` wartości równej 50 w pętli wewnętrznej powodowało wykonanie (`continue`) kolejnej iteracji dla zmiennej `i` pętli zewnętrznej (etykieta).

```
01: import java.util.Arrays;
02: import java.util.List;
03:
04: public class Main {
05:   public static void main(String[] args) {
06:      for( int i = 0 ; i < 100 ; i++ ) {
07:         for( int j = 0 ; j < 100 ; j++ ) {
08:            //TODO: Add a continuation of the outter loop
09:         }
10:      }
11:   }
12: }
```

■

Poprawnych odpowiedzi	*Ocena*
<0>	n/k
<1, 11>	2.0
<12, 13>	3.0
<14, 15>	3.5
<16, 17>	4.0
<18, 19>	4.5
<20>	5.0