Laboratorium 2

CEL: Typy danych. Metody. Cykl życia obiektów.

Polecenia:

Zadanie#1. Przeanalizuj kod źródłowy poniższych klas (umieszczonych w tym samym pakiecie):

```
class ScientificCalculator {
   private Calculator calc;

   ScientificCalculator ( ) {
       calc = new Calculator();
   }

   public int factorial(int b) {
     if(b<0) throw new Exception("Factorial not applicable for negatives");
     if(b==0) return 1;
     return b * factorial(b-1);
   }

   public int sumTo(int b) {
     if(b<0) throw new Exception("Sum To not applicable for negatives");
     if(b==0) return 0;
     return b + sumTo(b-1);
   }
}
```

```
class Calculator {
  private int version = 3;

  int add(int a, int b) {
     return a+b;
  }

  public int subtract(int a, int b) {
     return a-b;
  }

  private int multiply(int a, int b) {
     return a * b;
  }
}
```

Wskaż, które z poniższych działań będą poprawne:

a)	zamiana linii:
	`return b * factorial(b-1);`
	na linię:
	`return calc.multiply(b, factorial(b-1));`

b)	zamiana linii:
	`return b + sumTo(b-1);`
	na linię:
	`return calc.add(b, sumTo(b-1));`

c)	implementacja nowej metody w klasie `ScientificCalculator`:
	`public int add(int a, int b) {`
	`  return calc.add(a, b);`
	`}`

d)	implementacja nowej metody w klasie `ScientificCalculator`:
	`public int subtract(int a, int b) {`
	`  return calc.subtract(a, b);`
	`}`

e)	implementacja nowej metody w klasie `ScientificCalculator`:
	`public int getVersion() {`
	`  return calc.version;`
	`}`

■

Zadanie#2. Wskaż, które z poniższych stwierdzeń **NIE** są prawdziwe dla zmiennych finalnych:

a)	Nie można modyfikować nadanej wartości zmiennej finalnej
b)	Zgodnie z konwencją kodowania identyfikator klasowej zmiennej finalnej powinien być pisany wielkimi literami
c)	Domyślna wartość początkowa zmiennej finalnej jest równa 0
d)	Zmienna klasowa zadeklarowana jako finalna zajmuje inny segment pamięci niż zmienne niefinalne

■

Zadanie#3. Wskaż, które z poniższych metod nie posiadają zwracanego typu (w tym również z punktu widzenia bajtkodu Java):

a)	`public int myMethod(){ ... }`
b)	`public void myMethod(){ ... }`
c)	`public myMethod(){ ... } // bajtkod JVM`

■

Zadanie#4. Wskaż, które z poniższych zdań jest/są prawdziwe?

a)	Finalizacja jest zawsze uruchamiana przed odśmieceniem obiektu
b)	Finalizacja może zostać uruchomiona przed lub po odśmieceniu obiektu
c)	Finalizacja zostaje uruchomiona, kiedy obiekt staje się niedostępny
d)	Finalizacja umożliwia programiście zwolnienie pamięci przydzielonej obiektowi

■

Zadanie#5. Metodę dla poniższego fragmentu kodu:

```
public Image getImage(URL url, String name) throws IOException {
   try {
      // Something to do
```

```
      return getImage(new URL(url, name));
   }
   catch (MalformedURLException e) {
      return null;
   }
}
```

uzupełnij o komentarz javadoc (uwzględnij hipotetyczny opis, znaczniki: @param, @return, @throws). Przedstaw metodę uzupełnioną o komentarz javadoc oraz otrzymany wynik generowania HTML dla wykonanego opisu (screenshot).

■

Zadanie#6. Przeanalizuj poniższy kod.

```
1: public class Test {
2:     public static void main(String args[]) {
3:         method(0);
4:     }
5:
6:     public void method( int in ) {
7:         System.out.println( in );
8:     }
9:  }
```

Jaki wynik wykonania powyższego kodu zostanie wyświetlony na konsoli? Wyjaśnij przyczynę tej sytuacji.

■

Zadanie#7. Przedstaw, jaka jest postać znacznika javadoc opisującego autora z podaniem własnego imienia i nazwiska?

■

Zadanie#8. Przeanalizuj poniższy kod.

```
01: class Player {
02:     static int playerCount = 0;
03:     private String name;
04:
05:     public Player(String n) {
06:         name = n;
07:         playerCount++;
08:     }
09: }
10: public class PlayerTestDrive {
11:     public static void main(String[] args) {
12:         System.out.println(Player.playerCount);
13:         Player one = new Player("Tommy");
14:         System.out.println(Player.playerCount);
15:     }
16: }
```

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

a)	1 1
b)	0 0
c)	0 1
d)	1 0

■

Zadanie#9. Wskaż poprawne deklaracje lub definicje tablicy.

a)	`int[] []x[];`
b)	`int *x;`
c)	`int x[5];`
d)	`int[] x = {1,2,3};`
e)	`int x[];`
f)	`[]int x[];`

■

Zadanie#10. Przeanalizuj poniższy kod.

```
01: class Mixer {
02:     Mixer() {
03:     }
04:
05:     Mixer(Mixer m) {
06:         m1 = m;
07:     }
08:
09:     Mixer m1;
10:
11:     public static void main(String[] args) {
12:         Mixer m2 = new Mixer();
13:         Mixer m3 = new Mixer(m2);
14:         m3.go();
15:         Mixer m4 = m3.m1;
16:         m4.go();
17:         Mixer m5 = m2.m1;
18:         m5.go();
19:     }
20:
21:     void go() {
22:         System.out.print("hi ");
23:     }
24: }
```

Wskaż, jaki będzie wynik wykonania powyższego kodu?

a)	`hi`
b)	`hi hi`
c)	Błąd kompilacji
d)	`hi hi`, następnie błąd

■

Zadanie#11. Przeanalizuj poniższy kod.

```
public class Example {
    Example( int i ) {
        System.out.println("Constructor Example#" + i );
    }
    public static void main( String[] args ) {
        for( int i = 0; i < 10; i++ ) {
            new Example( );            // tutaj błąd!
        }
    }
}
```

Przedstaw dwa możliwe sposoby modyfikacji w powyższym kodzie konstrukcji obiektu `Example`, aby zapewnić jego poprawność.

■

Zadanie#12. Przeanalizuj poniższy kod.

```
public class Bird {
     Bird( String name ) {
        System.out.println("Bird Name = " + name);
     }

     Bird( ) {
        System.out.println("Constructor Bird.class");
        this( ”Starling” );
     }

     public static void main( String args[ ] ) {
        Bird a = new Bird( );
     }
}
```

Wyjaśnij przyczynę błędu w powyższym kodzie. Przedstaw najprostsze rozwiązanie błędu (kod całościowo i bez usuwania linii; możliwe jedynie przesunięcia instrukcji).

■

Zadanie#13. Przeanalizuj poniższy kod.

```
class Animal
{
     Animal( )
     {
        System.out.println( „Konstruktor Animal” );
     }
}
```

```
class Bird
{
     static Animal animal = new Animal( );
}

public class Example
{
     public static void main( String...args )
     {
        Bird bird1 = new Bird( );
        Bird bird2 = new Bird( );
     }
}
```

Wskaż, ile razy zostanie wyświetlony łańcuch tekstowy „`Konstruktor Animal`”. Wyjaśnij przyczynę z uwzględnieniem modułu ładującego JVM (ClassLoader).

■

Zadanie#14. Przeanalizuj poniższy kod.

```
public class Example
{
    public static void main( String[] args )
    {
        Example e1 = new Example( );
        // TODO: wyświetl nazwę typu (wyłącznie) dla e1
    }
}
```

W metodzie `main()` dopisz fragment kodu, w którym przy zastosowaniu obiektu klasowego zostanie wyświetlona na STDOUT nazwa typu ***(wyłącznie!)*** dla referencji o identyfikatorze `e1`.

■

Zadanie#15. Mając zmienne: `int i` oraz `long z` wskaż, które z poniższych instrukcji są poprawne?

a)	`i = z;`
b)	`i(long) = z;`
c)	`z = i;`
d)	`i = z(int);`
e)	`i = (int)z;`

■

Poprawnych odpowiedzi	*Ocena*
<0>	n/k
<1, 8>	2.0
<9, 10>	3.0
<11>	3.5
<12, 13>	4.0
<14>	4.5
<15>	5.0