

Obiekty

Pytania – Odpowiedzi

1. Spójrz na poniższy kod i odpowiedz na pytania w podpunktach.

```
Samochód s;
```

1.1. Czy powyższy kod tworzy obiekt?

Nie. Tworzy jedynie referencję/odwołanie do obiektu.

1.2. Czym jest `Samochód`?

Jest to typ obiektu.

1.3. Czym jest `s` ?

Jest to referencja/odwołanie do obiektu, mimo że obiekt jeszcze nie powstał, tj. nie został zainicjowany. `s` jest nazwą dla obiektu, który zostanie utworzony.

2. Spójrz na poniższy kod i odpowiedz na pytania w podpunktach.

```
Samochód s = new Samochód();
```

2.1. Czy powyższy kod tworzy obiekt?

Tak. `new` tworzy nowy obiekt danego typu. Tutaj tworzy nowy samochód, tj. nowy obiekt o typie `Samochód`.

2.2. Czy `s` jest obiektem?

Nie. `s` jedynie wskazuje na obiekt. Jest odwołaniem do obiektu. Jeden obiekt może mieć więcej niż jedną referencję, stąd `s` może być jedną z wielu referencji.

3. Co to jest zasięg?

Zasięg określa widoczność i czas życia zmiennych. Zasięg określa się za pomocą nawiasów klamrowych { }.

4. Czy poniższy kod się skompiluje?

```
public class Main {  
    public static void main(String args[]) {  
        int x = 4;  
        {  
            {  
                {  
                    int y = 5;  
                }  
                System.out.println(y);  
            }  
        }  
    }  
}
```

Nie, gdyż zmienna y jest poza zasięgiem funkcji drukowania.

5. Co to jest garbage collector i jak działa?

Odśmieca pamięć. Identyfikuje obiekty do których nie ma powiązanych referencji. Referencje mogą zostać przerwane jako rezultat wyjścia poza zasięg, w którym dana referencja istniała/została zdefiniowana.

6. Co to jest metoda w kontekście klasy?

Metoda to funkcja, która jest częścią klasy.

7. W poniższej metodzie wskaż typ zwracany, nazwę metody, argumenty oraz ciało.

```
int dodaj(int x, int y) {  
    return x+y;  
}
```

Typ zwracany – int
nazwa metody – dodaj
argumenty – int x, int y
ciało – return x+y

8. Do czego służy słowo kluczowe **import**?

Do importowania pakietów. Pakiety zawierają klasy.

9. Na co wskazuje użycie słowa kluczowego **static**?

Na to, że pole lub metoda nie jest przypisana do konkretnego obiektu oraz pozwala skorzystać z tych pól i metod bez tworzenia obiektu.

10. Jak jest najpopularniejsze użycia słowa **static**?

```
public class Main {  
    public static void main(String args[]) {  
        }  
}
```

Metoda **main** jest wywoływana bez tworzenia obiektu.

Przykłady

Przykład 1

- Napisz kod który stworzy instancję klasy Samochód.
- Używając instancji z podpunktu a), nadaj polu kolor wartość czerwony.
- Dopisz do klasy Samochod nowe pole metalik typu boolean.
- Zmień wartość pola metalik, tak by wskazywało na to, że samochód posiada kolor metalik.

```
public class Main {  
    public static void main(String args[]) {  
        Samochod s = new Samochod(); // (a)  
        s.kolor = "czerwony"; // (b)  
        s.metalik = true; // (d)  
    }  
}  
  
class Samochod {  
    String marka;  
    String kolor;  
    double dlugosc;  
    double szerokosc;  
    double predkosc;  
    boolean metalik; // (c)  
}
```

Przykład 2

Zainicjuj zmienną s wartością Witaj Swiecie

```
public static void main(String args[]) {  
    String s;  
    // String s = "Witaj Świecie";  
    // lub  
    // String s = new String("Witaj Świecie");  
}
```

Przykład 3

Jaką wartość wydrukuje poniższy kod?

```
public class Main {  
    public static void main(String args[]) {  
        int x = 2;  
        {  
            int y = 3;  
        }  
        System.out.println(y);  
    }  
}
```

Odp. Nie skompiluje się. y jest poza zasięgiem funkcji println.

Przykład 4

Zmodyfikuj poniższy kod, tak by typ zwracany funkcji dodaj był ustawiony na int.

```
class Liczydło {  
    dodaj(int x, int y) {  
        // int dodaj(int x, int y) {  
            return x+y;  
        }  
    }  
}
```

Przykład 5

Co wydrukują 3 funkcje println?

```
public class Main {  
    public static void main(String args[]) {  
        Samochod.kolor = "czarny";  
  
        Samochod s1 = new Samochod();  
        s1.kolor = "czerwony";  
  
        Samochod s2 = new Samochod();  
        s2.kolor = "niebieski";  
  
        System.out.println(Samochod.kolor);  
        System.out.println(s1.kolor);  
        System.out.println(s2.kolor);  
    }  
}  
  
class Samochod {  
    String marka;  
    static String kolor;  
    double dlugosc;  
    double szerokosc;  
    double predkosc;  
    boolean metalik;  
}
```

Odp. niebieski

niebieski

niebieski

Wynika to z faktu, że zmienna kolor jest static, a więc wartość jest współdzielona przez wszystkie obiekty. Do zmiennej jest dostęp bez tworzenia obiektu.

Pytania Prawda/Fałsz

1. Czy poniższy kod tworzy obiekt?

```
String s = "jakiś tekst";
```

2. Czy poniższy kod jest prawidłową konstrukcją i się skompiluje?

```
String s2 = new String("jakiś tekst");
```

3. Czy poniższy kod jest prawidłową konstrukcją i się skompiluje?

```
String s3 = new String('jakiś tekst');
```

4. Czy poniższy kod się skompiluje i uruchomi?

```
public class Main {  
    public static void main(String args[]) {  
        {  
            {  
                {  
                    System.out.println("Witaj Świecie");  
                }  
            }  
        }  
    }  
}
```

5. Nazwa klasy publicznej powinna odpowiadać nazwie pliku.

Pytania Prawda/Fałsz – Odpowiedzi

1. Prawda.

Typ String nie wymaga słowa new. Obiekt może być zainicjowany poprzez umieszczenie tekstu w cudzysłowie.

2. Prawda.

Tak. String można utworzyć tak, jak każdy inny obiekt za pomocą słowa new.

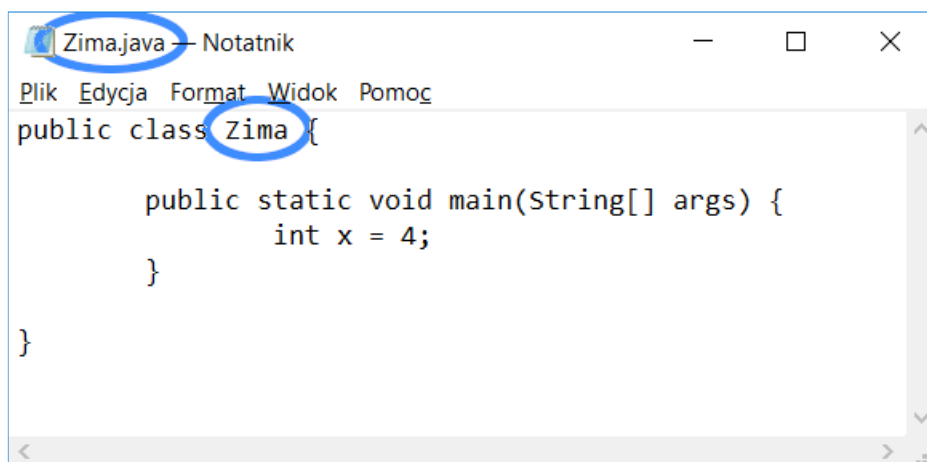
3. Fałsz.

W Pythonie pojedynczy cudzysłów jest dozwolony, ale Java wymaga podwójnego cudzysłowu.

4. Prawda.

{ } definiują kolejne zagnieżdżenia zasięgów. W tym przykładzie nie mają wpływu na wykonanie metody drukowania.

5. Prawda



```
Zima.java - Notatnik
Plik  Edycja  Format  Widok  Pomoc
public class Zima {
    public static void main(String[] args) {
        int x = 4;
    }
}
```