1. Kolejność inicjalizacji obiektu.

- pola statyczne
- deklaracje
- konstruktor
- instrukcje

Pola klasy są zerowane (referencje otrzymują wartość *null*), zmienne lokalne nie.

Tworzenie obiektu:

```
Object o = new Object();
```

Każda klasa automatycznie dziedziczy po klasie Object (otrzymuje jej właściwości).

- this – wskazanie na aktualny obiekt
- super – wskazanie na obiekt klasy bazowej (nadrzędnej)

Metody klasy Object:

.clone() – tworzy i zwraca kopię aktualnego obiektu

.equals(Object) – sprawdza, czy dany obiekt jest równy aktualnemu

.finalize() – metoda wywoływana przez Garbage Collector, gdy nie istnieje żadna referencja do aktualnego obiektu

.toString() – zwraca reprezentację tekstową obiektu

Usuwanie obiektów:

– brak destruktorów – pamięć odśmiecana automatycznie (Garbage Collector)

– gdy klasa rezerwuje pamięć w sposób niestandardowy, należy zwolnić ją ręcznie

```
protected void finalize() throws Throwable {
      //...
      super.finalize();
}
```

– wymuszenie uruchomienia Garbage Collector

```
System.gc();
System.runFinalization();
```

Tablice:

- tworzenie tablicy jednowymiarowej typu int

```
int[] tab = new int[10];
```

– tworzenie tablicy wielowymiarowej

```
int[][] tab = new int[10][4];
```

– tworzenie i inicjalizacja zawartości tablicy

```
int[] tab = { 5, 3, 8, 2, 7 };
```

– liczba elementów tablicy

```
tab.length
```

– odwołanie do i-tego elementu tablicy

```
tab[i]
```

2. Konstruktor.

Jest specjalnym rodzajem metody. Jego nazwa musi być taka sama jak nazwa klasy, w której jest zadeklarowany (po tym kompilator poznaje, że ma do czynienia z konstruktorem, a nie zwykłą metodą). Dla konstruktora nie podaje się typu wyniku. Liczba parametrów konstruktora może być dowolna. Jeśli sami nie zdefiniujemy konstruktora, to kompilator sam wygeneruje bezargumentowy *konstruktor domyślny*. Dzieje się tak tylko wtedy, jeśli autor klasy nie zdefiniuje żadnego własnego konstruktora.

```
class Osoba {
	String imie;
	String nazwisko;

	Osoba(String imię, String nazwisko) {
		this.imie = imie;
		this.nazwisko = nazwisko;
	}

	String imie() {
		return imie;
	}

	String nazwisko() {
		return nazwisko;
	}

}
```

W treści konstruktora użyliśmy słowa kluczowego *this*. Oznacza ono obiekt, na rzecz którego wykonywana jest metoda. Dzięki użyciu *this* możemy łatwo wskazać, czy chodzi nam o zmienną obiektową (*this.imie*) czy o parametr metody (*imie*).

Tworzenie obiektu:

```
Osoba o = new Osoba(); // Po dodaniu konstruktora powoduje błąd kompilacji
Osoba o = new Osoba("Jan","Kowalski"); // Poprawne
```

Przykład:

```
class Telewizor {
	public String marka;
	public int cale;
```

```
	// konstruktor
	public Telewizor(String marka, int cale) {
		this.marka = marka;
		this.cale = cale;
	}

	// przesłonięta metoda
	public String toString() {
		return "Telewizror: " + marka + " " + cale + "\"";
	}
}

public class Test {
	public static void main(String[] args) {
		System.out.println(new Telewizor("SAMSUNG", 42));
		// automatyczna konwersja typu - wywołanie metody toString
	}
}

Telewizror: SAMSUNG 42"
```

Zadanie:

Zaimplementować aplikację z wykorzystaniem klas wyposażonych w konstruktory (kilka wersji konstruktorów). Przesłonić metodę *toString*.