1. Typy.

Java jest językiem programowania z silnym systemem kontroli typów. To oznacza, że każda zmienna, atrybut czy parametr ma zadeklarowany typ. Kompilator wylicza typy wszystkich wyrażeń w programie i sprawdza, czy wszystkie operatory i metody są używane zgodnie z ich deklaracjami, czyli z argumentami odpowiednich typów. Także elementy każdej instrukcji muszą mieć właściwe typy, np. warunek w pętli **while** musi być wyrażeniem o wartości typu logicznego. Specyficzną cechą Javy jest to, że typy w tym języku są podzielone na dwie kategorie:

- typy pierwotne,
- typy referencyjne.

Typy pierwotne to grupa ośmiu typów zawierających wartości proste. Tymi typami są:

- typ wartości logicznych: *boolean*,
- typy całkowitoliczbowe: *byte*, *short*, *int*, *long*, *char*,
- typy zmiennopozycyjne: *float*, *double*.

Typy referencyjne dzielą się z kolei na następujące kategorie:

- typy klas,
- typy interfejsów,
- typy tablic.

Wartościami typów referencyjnych są referencje (w pewnym uproszczeniu można o nich myśleć jako o wskaźnikach) do obiektów lub wartość *null*.

Typy obiektowe opakowujące typy proste:

- Boolean (boolean)
- Character (char)
- Byte (byte)
- Short (short)
- Integer (int)
- Long (long)
- Float (float)
- Double (double)
- Void (void)
- BigInteger - typ całkowity dowolnej precyzji
- BigDecimal - typ stałoprzecinkowy dowolnej precyzji
- String (char[]) - łańcuch tekstowy

2. Obiekty.

Przez obiekt rozumie się w Javie dynamicznie stworzony egzemplarz jakiejś klasy lub dynamicznie stworzoną tablicę. Żeby to zrównanie egzemplarzy klas i tablic uprawomocnić zadbano, by zarówno egzemplarze klas jak i tablice rozumiały wszystkie metody z klasy *Object*.

3. Zmienne.

Zmienne są (zwykle) nazwanymi pojemnikami na pojedyncze wartości typu z jakim zostały zadeklarowane. Zmienne typów pierwotnych przechowują wartości dokładnie tych typów, zmienne typów referencyjnych przechowują wartość *null* albo referencję do obiektu typu będącego zadeklarowanym typem zmiennej bądź jego podklasą. Każda zmienna musi być zadeklarowana. Z każdą zmienną związany jest jej typ podawany przy deklaracji zmiennej. Typ ten jest używany przez kompilator do sprawdzania poprawności operacji wykonywanych na zmiennych. W Javie każda zmienna musi być zainicjowana przed pierwszym pobraniem jej wartości. Wymusza to kompilator Javy. W Javie przyjęto, że takie zmienne albo będą jawnie inicjowane przez programistę albo będą miały nadaną automatycznie wartość początkową. Zasady są tu następujące: zmienne klasowe, egzemplarzowe i elementy tablic są inicjowane wartościami domyślnymi, zaś zmienne lokalne muszą być jawnie zainicjowane przez programistę.

4. Konwencja nazewnicza.

Konwencje kodowania w Javie zalecają pisanie nazw klas i interfejsów z wielkiej litery, nazw zmiennych, parametrów i metod z małej. W nazwach wielosłowowych kolejne słowa piszemy z wielkiej litery (bez podkreślników, jest to tzw. notacja CamelCase). Nazwy pakietów piszemy zawsze wyłącznie małymi literami, nazwy stałych (static final) wyłącznie wielkimi.

5. Podstawowe konstrukcje języka Java.

Klasa: NazwaKlasy.java

```
public class NazwaKlasy {
	//pole (zmienna/stała/referencja)
	public int i;
	//konstruktor (domyślny/sparametryzowany)
	public NazwaKlasy() { }
	//metoda
	public void nazwaMetody() { }
	//metoda main - punkt wejścia programu
	public static void main(String[] args) {
```

```
            //...
        }
}
```

Klasa publiczna determinuje nazwę pliku Java. Konstruktor ma identyczną nazwę jak klasa.

Przykład:

```
/**Pakiet
 *
 */
package com.labjava;
/**
 * @author Anonim
 * @version 1.1
 * @see Object
 *
 */
public class Zarowka {
	/**
	 * zmienna określająca moc żarówki
	 */
	public int moc;
	/**
	 * Konstruktor klasy Zarowka
	 *
	 * @param moc - argument określający moc żarówki
	 */
	Zarowka(int moc) {
		this.moc = moc;
	}
	/**
	 * metoda zapalająca żarówke
	 */
	public void zapal() {
		System.out.println("Żarówka świeci z mocą " + moc + "W");
	}
	/**
	 * metoda wyłączająca żarówkę
	 */
	public void zgas() {
		System.out.println("Żarówka nie świeci");
	}
	/**
	 * @param args - argumenty przekazywane do programu z linii komend
	 */
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Zarowka z = new Zarowka(100);
		z.zapal();
		z.zgas();
	}
}
```

Wyrażenia:

i++ – postinkrementacja

++i – preinkrementacja

i += 2 – przypisanie złożone (i = i + 2)

Kolejność operatorów:

Typ operatora	Operatory
Jednoargumentowe	+ - ++ --
Arytmetyczne oraz przesunięcia	* / % + - << >>
Relacyjne	> < >= <= == !=
Logiczne i bitowe	&& || & | ^
Warunkowy trójargumentowy	A > B ? X : Y
Przypisania	= (oraz przypisania złożone)

Instrukcje sterujące:

- pętla while

```
while (i < 8) { ... }
```

- pętla do-while

```
do { } while (i < 8);
```

- pętla for

```
for (int i = 0; i < 8; i++) { }
```

- warunek if-else

```
if (i < 8) { } else { }
```

- wybór switch

```
switch (test) {
  case 1 : { } break;
  case 2 : { } break;
  default : { }
 }
```

– instrukcja ***break*** - przerywa iterację (wyjście z pętli)

– instrukcja ***continue*** - przechodzi do następnej iteracji (wraca na początek pętli)

– instrukcja ***continue*** [etykieta] - skacze do etykiety i ponownie wchodzi do pętli

– instrukcja ***break*** [etykieta] - przechodzi poza koniec pętli oznaczonej etykietą

– instrukcja ***return*** – natychmiastowy powrót z metody

– pętla nieskończona

```
  for (;;){}
while (true) {}
```

Składnia wszystkich instrukcji sterujących jest zgodna z językiem C.

Porównywanie:

– typy proste

• operator relacji równości zwraca ***true*** lub ***flase*** na podstawie porównania wartości zmiennych (i==n)

– obiekty

• operator relacji porównuje referencje obiektów

• aby porównać obiekty (ich właściwości) należy użyć metody equals(Object) z klasy Object.

Rzutowanie:

– niejawne (automatyczne)

• promocja typu prostego

```
int i = 200;
long k = i;
```

• rzutowanie „w górę” obiektów

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

– jawne (z określeniem typu rzutowania w nawiasach)

• rzutowanie typów prostych

```
int j = (int) k;
```

• rzutowanie obiektów

```
String s = (String) o;
```

Mechanizmy wewnętrzne Javy:

Podczas operacji arytmetycznych następuje automatyczna promocja typu (rozszerzenie typu zmiennych) do największego występującego w operacji matematycznej.

W przypadku operacji logicznych następuje skracanie obliczeń wyrażeń logicznych, gdy osiągnięty rezultat nie może ulec zmianie (np. *false* && *true*, *true* || *false*).

Przykład:

```
public class Test {
	/**
	 * @param args
	 *            argumenty przekazywane do programu z linii komend
	 */
	public static void main(String[] args) { // args - tablica argumentow
		if (args.length > 0)
			System.out.println("Argument: " + args[0]);

		else
			System.out.println("Losowa liczba z przedziału <0,100): "
					+ (int) (Math.random() * 100));
			// (int) - konwersja typu
	}
}
```

Zadanie:

Za pomocą instrukcji sterujących oraz pętli napisz aplikację symulującą działanie menu telefonu komórkowego. Przejście po menu ma być realizowane za pomocą generatora liczb pseudolosowych. W pierwszym poziomie menu należy wykorzystać instrukcję switch, w drugim poziomie instrukcję if-else natomiast w trzecim ponownie instrukcję switch. Symulacja działania menu odbywać się ma w co najmniej 10 przebiegach, zrealizowane z wykorzystaniem pętli for.