Vježbe 1 - upoznavanje s razvojnom okolinom i izrada prvog *Java* programa

Matej Mihelčić

Prirodoslovno-matematički fakultet, Sveučilište u Zagrebu

matmih@math.hr

11. listopada, 2022.

Java development kit (JDK)

Java runtime environment - JRE je platforma potrebna za **pokretanje** *Java* programa na podržanom uređaju. Sadrži *Java virtualni stroj*, ključne klase *Java* platforme i popratne biblioteke *Java* platforme[1]. Podržava *Windows* 32/64-bit, *Linux* 32/64-bit, *Mac OS X*, *SOLARIS SPARC*/x64.

Za **razvoj** *Java* aplikacija potrebno je instalirati *Java development kit - JDK*. Sadrži alate za razvoj i testiranje *Java* aplikacija[2]. Postoji i besplatna (eng. open source) implementacija *Java JDK* koja se zove *OpenJDK*[3]. Podržava *Windows* 64-bit, *Linux* 64-bit, *Linux* 64-bit *ARM* i *Mac OS*.

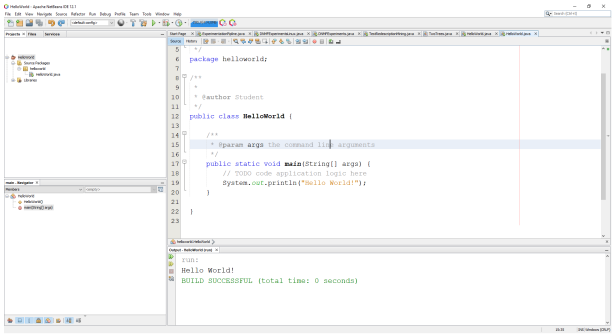
[1] `https://www.java.com/en/download/manual.jsp`
[2] `https://www.oracle.com/java/technologies/downloads/`
[3] `https://openjdk.java.net/`

Razvojna okolina

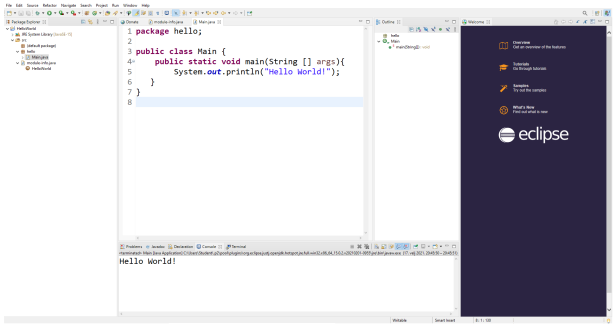
Softver za razvoj *Java* aplikacija koristeći instalirani *Java JDK*.

Primjer (tri najpoznatija integrirana okruženja za razvoj (eng. Integrated Development Environment - IDE):

- *NetBeans*[4]
- *Eclipse*[5]
- *IntelliJ*[6]



NetBeans



Eclipse

IntelliJ

[4] https://netbeans.apache.org/
[5] https://www.eclipse.org/ide/
[6] https://www.jetbrains.com/idea/?fromMenu

Struktura *JAR* dokumenta

JAR je format baziran na *ZIP* formatu (*ZIP* koji potencijalno sadrži poseban `META-INF` direktorij). Koristi se za agregiranje više dokumenata u jedan dokument. *JAR* je osnovni element za izgradnju aplikacija i ekstenzija.

`META-INF` direktorij sadrži konfiguracijske podatke paketa i ekstenzija, uključujući podatke za verzioniranje, proširenja, usluge i sigurnosnu verifikaciju.

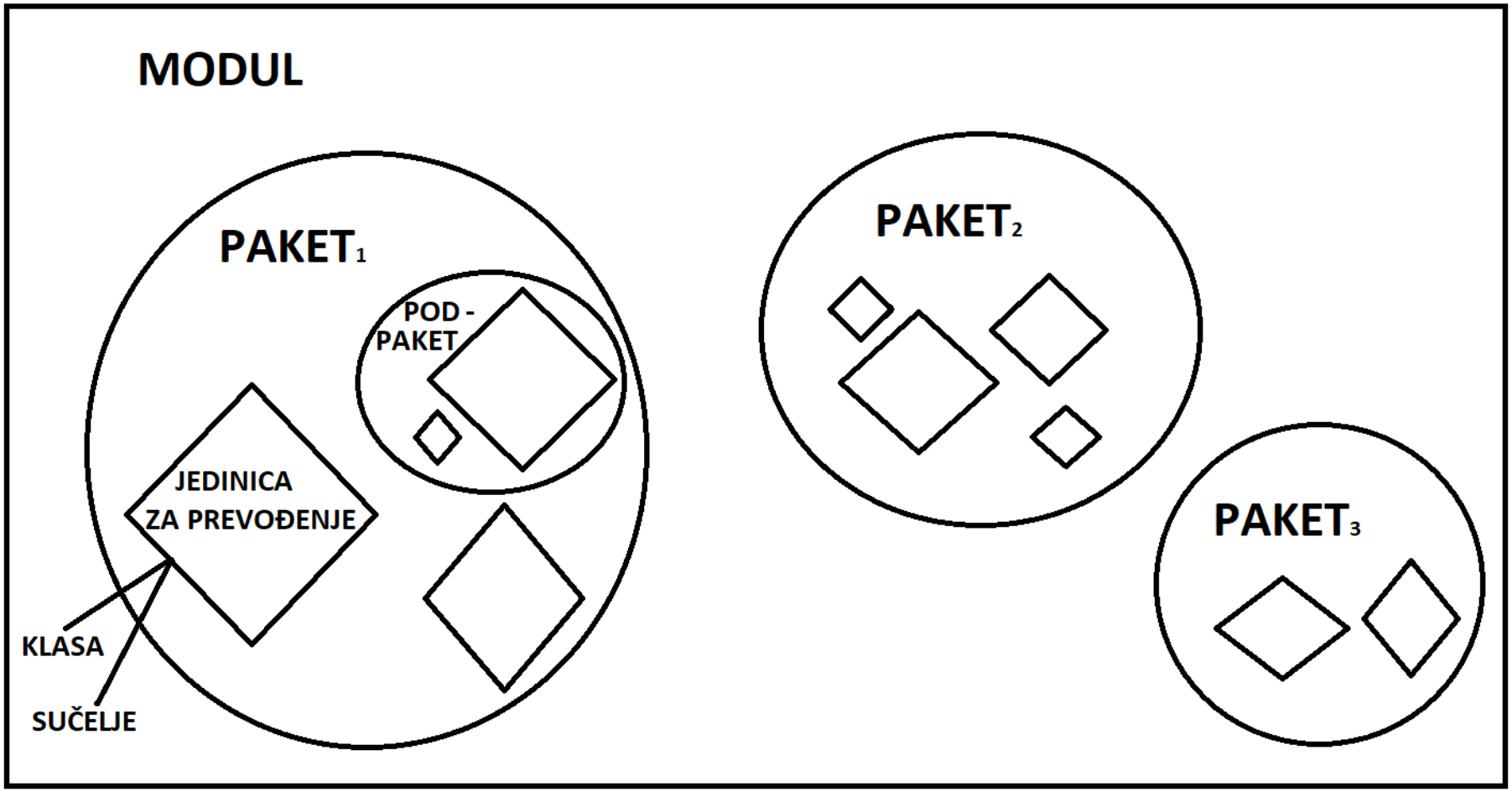
`META-INF` direktorij može sadržavati sljedeće dokumente:

- `MANIFEST.MF` - koristi se za definiranje ekstenzija i informacija vezanih uz pakete
- `INDEX.LIST` - sadrži informacije o lokaciji paketa definiranih u aplikaciji ili proširenjima.
- `x.SF` - dokument koji sadrži digitalni potpis *JAR* dokumenta.
- `x.DSA` - sadrži digitalni potpis dokumenta `x.SF`
- `services/` - sadrži sve konfiguracijske datoteke korištenih pružatelja usluga.

Primjer MANIFEST dokumenta

```
Manifest-Version: 1.0
Ant-Version: Apache Ant 1.10.4
Created-By: 15.0.1+9-18 (Oracle Corporation)
Class-Path:
X-COMMENT: Main-Class will be added automatically by build
Main-Class: helloworld.HelloWorld
```

Struktura *Java* programa



Jedinica za prevođenje je jedna datoteka izvornog koda (.java datoteka) koja sadrži definiciju **klase** ili **sučelja**.

Struktura *Java* programa

Jedinice za prevođenje se grupiraju u **pakete**. Paketi mogu sadržavati i **pod-pakete** koji su po strukturi opet paketi ali niže razine. **Paketi ne smiju sadržavati dva člana istog imena ili dolazi do greške pri prevođenju!**

Paketi imaju imena iako je **moguće** prevoditi jedinicu za prevođenje i **bez deklaracije paketa**.

Iz zadanog paketa možemo koristiti samo `public` klase i sučelja definirane u **nekim drugim paketima** (više detalja u kasnijim predavanjima i vježbama).

Klase i sučelja iz pod-paketa **nisu standardno dostupni** unutar paketa.

Konvencije za imenovanje objekata u *Java* programima

- Ime paketa se **uvijek** piše **malim slovima** (eng. *lowercase*). Ovo vrijedi i kada se ime paketa sastoji od više riječi.
- Imena klasa i sučelja se pišu na način da je **prvo slovo svake riječi uvijek veliko** (eng. *UpperCamelCase*). Imena klasa i sučelja su obično **imenice**.
- Imena metoda se pišu na način da **uvijek počinju malim slovom** a **za sve sljedeće riječi je prvo slovo veliko** (eng. *lowerCamelCase*). Imena metoda su obično glagoli.
- Imena instanci, klasa i konstanti klasa se također pišu koristeći *lowerCamelCase* notaciju. Imena varijabli ne bi trebala počinjati znakom '_' ili '$' (iako su formalno dozvoljeni).
- Imena varijabli koje su po deklaraciji konstante klasa ili *ANSI* konstante, pišu se svim velikim slovima na način da su riječi odijeljene znakom '_' (npr. `MIN_WIDTH`, `NUM_THREADS`) itd.

Konvencije za imenovanje objekata u *Java* programima

Preporučeno koristiti deskriptivna imena za pakete, klase, metode i varijable. Na taj način je lakše **razumijeti kod** (pogotovo osobama koje ga prvi puta gledaju) i **locirati dijelove koda s određenom funkcionalnosti**. Možemo dodati i prefiks imenima varijabli koji označava tip (i-integer, s-string itd.). Na taj način ne moramo tražiti deklaracije varijabli u kodu. Imena varijabli od jednog znaka (i, j, k itd.) se trebaju izbjegavati osim za **privremene varijable** (petlje, dimenzije, indeksi itd.).

Zadatak 1

- Napravite program koji sadrži paket pod imenom *prvi*. Unutar njega napravite izvornu datoteku *PrviProgram.java* koja sadrži kod glavne funkcije koja ispisuje poruku *"Prvi program u Javi!"*.
- Deklarirajte po jednu varijablu svakog osnovnog tipa *short*, *byte*, *boolean*, *int*, *float*, *double*, *char*. Deklarirajte varijablu cjelobrojnog tipa koja ima širi raspon od *int*. Pridijelite varijablama maksimalnu/minimalnu vrijednost određenog tipa. Probajte tim vrijednostima dodati/oduzeti broj 1. Što se događa? Probajte podijeliti varijablu tipa *float* s 0, što se događa? Što je ostatak pri dijeljenju broja *a*, brojem *b*, gdje su oba tipa *float* ili *double*?
- Deklarirajte i inicijalizirajte jedan broj u oktalnom i jedan u heksadekadskom sustavu i ispišite njihove vrijednosti.
- Zapakirajte dvije varijable primitivnog tipa *int* u odgovarajuću klasu *Integer*. Čemu to služi? Da bi mogli koristiti primitivne tipove kod ili s objektima koji ih ne podržavaju.

Zadatak 1

- Isprobajte implicitnu i explicitnu konverziju između tipova.
- Isprobajte operatore navedene na predavanjima. Koja je razlika između operatora $>>>$ i $>>$? Što se događa ako ga upotrijebimo na negativnim brojevima?
- Kako možemo računati s jako velikim brojevima? Korištenjem klasa `BigInteger` i `BigDecimal` iz paketa `java.math`. Ove klase su ograničene jedino količinom memorije koju može koristiti *Java virtualni stroj*.

Zadatak 2

- Definirajte novu izvornu datoteku imena *Tekst.java* u istom paketu *prvi*. Definirajte dvije varijable tipa *String* i isprobajte: a) inicijalizaciju, b) konkatenaciju (rezultat se sprema u prvu varijablu), c) izbacite iz prvog stringa sve samoglasnike, d) ispišite zadnji znak drugog stringa, d) ispitajte imaju li stringovi jednake znakove od 2-8 pozicije, e) napravite string koji nastaje k-uzastopnom konkatenacijom prvog stringa sa samim sobom, f) pronađite zadnje pojavljivanje stringa *"aba"* u drugom stringu, g) promijenite prvi string tako da sadrži samo mala slova.
- Definirajte varijablu koja sadrži blok teksta te ispišite njen sadržaj u konzolu. Definirajte novu varijablu koja sadrži neki novi (drugačiji) blok teksta. Kreirajte blok teksta koji sadrži informacije iz oba bloka gdje je tekst različitih blokova razmaknut s dva prazna retka. Ispišite novo dobiveni blok teksta.