Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/01BE33Urosevic

NOVI STANDARDI PROGRAMSKOG JEZIKA JAVA

THE NEW STANDARD IN PROGRAMMING JAVA LANGUAGE

Dejan Urošević, *Fakultet tehničkih nauka, Novi Sad*

Oblast – RAČUNARSTVO I AUTOMATIKA

Kratak sadržaj – *Tema rada jeste predstavljanje novina i poboljšanja novih verzija programskog jezika Java. U okviru rada predstavljene su novine u verzijama 8, 9 i 10. Sadržaj rada pored opisa novina sadrži i jednostavne primere u kojima se iste te novine i poboljšanja koriste.*

Ključne reči: *Java, lambda izrazi, Stream, Asinhroni pozivi, API, JavaScript, garbage collector, bytecode, modularni sistem, var, sertifikat*

Abstract – *The aim of this paper is to present the new add-ons and improvement in the newest version of the Java programming language. The content of this paper is an improvement in versions 8, 9 and 10. Beside the description this paper contains examples in which, the new add-ons and improvement are used.*

Keywords: *Java, lambda expression, Stream, asynchronous call, API, JavaScript, garbage collector, bytecode, modular system, var, certificates*

1. UVOD

Java [1] predstavlja programski jezik razvijen početkom 1990-ih godina od grupe ljudi iz *Sun Microsytem* firme koju je predvodio Džejms Goslin.. Prva verzija Jave nije bila dovoljno dobra za razvoj nekih ozbiljnijih aplikacija i tim je morao da nastavio sa razvojem kako bi napravili konkurentan proizvod. Programski jezik Java danas je jedan od najpopularnijih i najviše korišćenih, a sve to zahvaljujući njegovim karakteristikama opisanim u narednom delu rada.

2. KARAKTERISTIKE PROGRAMSKOG JEZIKA JAVA

2.1 Jednostavnost

Jednostavnost se dobila pojednostavljenjem programskog jezika C++ [2]. Iz razloga što su fajlovi bili dovoljno mali bila je potrebna prosečna mašina za izvršavanje aplikacija.

2.3 Distribuiranost

Uz pomoć Java aplikacija veoma lako se pristupa mreži i objektima na mreži, a zamorni zadaci poput otvaranja mrežne konekcije veoma su pojednostavljeni.

2.4 Robusnost

Java je programski jezik koji je napravljen da bude pouzdan na mnogo načina. Proverava ispravnost koda kako u ranoj fazi, tako i prilikom izvršavanja i omogućava ranu eliminaciju grešaka

2.5 Bezbednost

Napadi kao što su prekoračenje izvršnog steka, pristup memoriji izvan dela dodeljenog procesu ili čitanje i upisivanje datoteka bez dozvole bili su onemogućeni od samog početka. Tokom vremena dodati su i dodatni bezbednosni mehanizmi kao što je digitalni potpis klase tako da ako neko veruje autoru iste može omogućiti istoj veće privilegije na svojoj mašini.

2.6 Neutralna arhitektura

Java programski jezik se ne prevodi direktno u mašinski nego prvo se prevodi u *bytecode* [3], a ova funkcionalnost je dopustila da bude nezavistan od bilo koje arhitekture, a i dodatno je povećana bezbednost.

2.7 Prenostivost

Za razliku od drugih programskih jezika kod Jave ne postoje aspekti koji su zavisni od implementacije. Aplikacije koje se prave pomoću Jave mogu da se pokrenu na bilo kom operativnom sistemu.

2.8 Treba da se interpretira

Kako bi se omogućila prenosivost i nezavisnost Java aplikacija od operativnog sistema inženjeri su napravili Java interpreter. Java interpreter predstavlja alat koji prevodi *bytecode* u mašinski kod.

2.9 Visoke performanse

Danas se *bytecode* se prevodi u letu, delovi koda se prevode u mašinski kod prilikom izvršavanja, i time se postižu performanse koje su konkurentne sa tradicionalnim kompajliranjem, a nekada su čak i bolje.

2.10 Višenitan

Lakoća višenitnog programiranja predstavlja jedan od glavnih razloga zašto se mnogi odlučuju da koriste Javu prilikom pravljenja serverskih aplikacija. Kod Jave, kod koji vrši višenitno pozivanje je isti svuda, dok se višenitna implementacija prebacuje na platformu na kojoj je aplikacija pokrenuta ili se koristi posebna biblioteka.

2.11 Dinamičan

Ova karakteristika omogućava dodavanje koda prilikom izvršavanja, pruža potpuni uvid u strukturu i ponašanje objekata neke aplikacije kao i prikupljanje raznih informacija.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Aleksandar Kupusinac.

3. JAVA 8

U martu 2014. godine predstavljena je 8. verzija, ujedno i poslednja koja ima dugoročnu podršku koja je planirana sve do 2025. godine. Ova verzija donela je dosta poboljšanja od kojih će neka biti nabrojana i ukratko opisana u ovom radu.

3.1 Lambda izrazi

Lambda izrazi [4] predstavljaju novi, a ujedno i najznačajniji dodatak programskom jeziku. Sintaksa lambda izraza je sledeća **parametri -> logika,** sve što je potrebno jeste da se napišu argumenti i logika koja je predstavljena kroz funkciju**.** Kako je Java tipiziran jezik promenljiva koja predstavlja lambda izraz mora imati neki tip kao vrednost. Tip koji se dodeljuje promenljivoj predstavlja funkcionalni interfejs koji je potpuno novi koncept uveden od verzije 8 i sadrži jednu i samo jednu apstraktnu metodu koja se menja prilikom implementiranja lambda izraza. U okviru ove novine dodata je i nova anotacija *@FunctionalInterface*. Prilikom pisanja promenljivih u okviru lambda izraza moraju se poštovati ista pravila kao i za ugnježdene blokove koda. Unutar tela funkcije ne mogu se naći ista imena promenljivih kao i u prethodnom delu koda, ako se tako nešto desi kod neće biti ispravan i kompajler će prikazati sintaksnu grešku. U koliko se želi pozvati samo jedna metoda u okviru lambda izraza inženjeri su napravili prečicu čija sintaksa je sledeća **objekat :: metoda**. Ovakav način pozivanja metoda može da se koristi nad statičkim metodama, metodama nekog objekta i prilikom kreiranja novog objekta uz pomoć ključne reči *new*.

3.2 Default i Static metode

U prethodnim verzijama Jave prilikom kreiranja interfejsa nije bilo moguće kreirati i metode koje imaju implementaciju. Implementirane metode u okviru intefejsa se nazivaju *default*-ne [5]. Ukoliko se implementiraju dva ili više interfejsa koji sadrže identične *default* metode prilikom poziva doći će do kolizije iz razloga što prevodilac neće znati koju tačno da pozove. Za rešavanje ovakvog problema postoje dva načina. Prvi način je da se metoda izmeni i postavi joj se nova logika, a kod drugog načina eksplicitno se navodi od kojeg interfejsa će se metoda pozvati.

3.3 Novi API za vreme

Raditi sa vremenom u Javi nikada nije bilo jednostavno, stari API je imao dosta grešaka, za neke stvari bilo je potrebno iskucati dosta linija koda kako bi sve funkcionisalo. U novom API-ju predstavljene su nove klase koje se mogu podeliti na one koje koriste vremensku zonu i one gde to nije potrebno. *LocalDate*, *LocalTime*, *LocalDateTime* predstavljaju nove klase gde se ne koristi vremenska zona, dok *ZonedDateTime* predstavlja klasu koja omogućava korišćenje vremenskih zona. Novim klasama omogućeno je lakše i jednostavnije manipulisanje sa vremenom, a pored toga nove klase su dosta otpornije na greške. U novoj verziji predstavljene su i klase *Period* i *Duration* uz pomoć kojih se predstavlja vreme koje je razumljivije čoveku. *Period* reprezentuje razliku između dva vremena u danima, mesecima ili godinama, dok *Duration* reprezentuje vrednost zasnovanu na nekom vremenu. Klasa *TemporalAdjusters* se koristi kako bi se izvele komplikovanije matematičke operacije. Pored klasa gde se ne koristi vremenska zona uvedene su i klase gde je omogućeno korišćenje istih. Klasa *java.time.ZonedId* može da se koristi sa fiksnim vrednostima za neku vremensku zonu ili navođenjem geografskog regiona. Ovom klasom se rešava problem kao što je letnje računanje vremena.

3.4 Nova OPTIONAL klasa

Nova Java dolazi sa *Optional* klasom koja se nalazi u paketu *java.util*. Pre su programeri morali da pišu dosta koda kako bi proverili da li je neki objekat jednak *null* da bi mogli da vrše neke operacije nad njim, a da te operacije ne izazovu neku grešku koja će dovesti do prestanka rada aplikacije.

3.5 Nashron, novi alat za JavaScript

U novoj verzije Jave zamenjen je stari JavaScript [6] alat sa novim, a u okviru novog dodate su i neke nove komande kao sto je *jjs*. Nova komanda omogućava da se JavaScript kod izvršava u okviru komandne linije. *Neshorn* [7] može da se koristi i u okviru Java koda gde uz pomoć *ScripEngineManager* klase JavaScript kod može da se poziva i interpretira.

3.6 Stream klasa

Omogućeno je da se operacije kao što su filtriranje i iteriranje kroz kolekcije obavljaju brže, pregledniji kod, uz pomoć stream API-ja omogućeno je i paralelno izvršavanje mnogih operacija koje do sada nisu bile moguće. Korišćenje Stream [8] API-ja u kombinaciji sa lambda izrazima podseća na funkcionalno programiranje koje se sve više koristi u okviru objektno orijentisanog programiranja.

3.7 Asinhroni pozivi

Od verzije 5 moguće je koristiti asinhrone pozive [9], ali je dosta teško raditi sa njima. U verziji 8 predstavljen je *CompletableFuture* interfejs i on predstavlja potpuno novi koncept koji objedinjuje *Future* i *CompletionStage* interfejse. *CompletionStage* predstavlja obećanje da će se poziv izvršiti, dobra stvar kod ovog interfejsa je što pruža veliki broj metoda nad kojima se može implementirati logika u odnosu na vrstu odgovora koji se dobije. *CompletableFuture* dozvoljava kreiranje i izvršavanje velikog broja asinhronih poziva, a da se pri tome ne blokira aplikacija.

3.9 Novi Base64 koder i dekoder

Klasa *Base64* [10] bila je dostupna i u prethodnim verzijama Jave, ali nije dolazila u okviru samog programskog jezika i bilo je potrebno dodati određene zavisnosti, odnosno *jar* fajlove koji su sadržali ovu klasu. Od verzije 8 ovaj, mali, ali veoma bitan API postao je sastavni deo Jave i sva njegova funkcionalnost je postala dostupna programerima u okviru paketa *java.utils.*

4. JAVA 9

U septembru 2017. godine *Oracle* je predstavio novu verziju Jave. Za razliku od prethodne verzije ova je zamišljena kao verzija u kojoj će se pokriti neki nedostaci

koji su naknadno primećeni, ali neće biti dugoročne podrške kao kod verzija 8.

4.1 G1 garbage collector

Java ima četiri *garbage collector-a* [11], verzija 8 kao podrazumevani koristi *Parallel/Throughput Collector*. U novoj verziji Jave kao podrazumevani *garbage collector* postavljen je G1 koji je prethodno predstavljen u 7 verziji programskog jezika Java. G1 je kreiran sa namerom da podrži *heap* veći od 4GB i samim tim poveća performanse, pored povećanja performansi trebalo je smanjiti broj pauza u odnosu na prethodni.

4.2 Novi HTTP 2.0 klijent

Novi API predstavljen je u okviru *incubator module-a* [12] što znači da je ovaj dodatak još uvek na ispitivanju i na osnovu povratnih informacija od klijenata biće odlučeno da li će biti i zvanično uključen u sledeću verziju Jave. HTTP 2.0 koristi *Build pattern* [13] za kreiranje zahteva. Prethodno navedeni šablon omogućava da se lako i brzo kreiraju zahtevi i više nije potrebno koristiti *InputStream* ili *Reader*. Najznačajnije klase su *HttpRequest, HttpResponse* i *HttpClient.*

4.3 Privatne metode u okviru interfejsa

Kako ne bi dolazilo do redundantnosti koda i određena logika se pisala više puta, u okviru verzije 9 omogućeno je kreiranje privatnih metoda u okviru interfejsa. Ove metode mogu da se koriste u okviru *default*-nih metoda.

4.4 API za slike sa više rezolucija

U okviru paketa *java.awt.image* kreatori Jave predstavili su novi API koji omogućava da se slike različitih rezolucije objedine u okviru jednog objekta. Klasa *MultiResolutionImage* predstavlja objekat u kojem su smeštene slike.

4.5 Modularni sistem – *Jigsaw* projekat

Modularni sistem predstavlja jedan od najvažnijih dodataka 9. verziji Jave. Jedan od glavnih razloga za uvođenje modularnosti jeste mogućnost pokretanja modularne Java Virtualne Mašine na različitim uređajima koji imaju manje memorije. Modul čini naziv, javno dostupni delovi i zavisni delovi. Prilikom kreiranja projekta koji će predstavljati modul obavezno je kreirati *module-info.java.* Sadržaj prethodno navedenog fajla je sledeći *module ime-foldera,* ime foldera predstavlja ime novog modula i po konvenciji obično se uzima ime foldera u kome se fajl nalazi.

4.6 JShell komanda (REPL)

Mnogi moderni jezici imaju funkcionalnost da u okviru terminala pišu određeni kod i izvrše ga bez prethodnog kompajliranja. Sa devetom verzijom Jave ova mogućnost je uvedena uz pomoć *JShell-a* koji predstavlja alat koji omogućava REPL funkcionalnost. *JShell* se pokreće u okviru terminala komandom *jshell.*

4.7 Dijamant operator dozvoljen za anonimne klase

U prethodnim verzijama nije bilo moguće pravljenje anonimnih klasa uz pomoć operatora dijamant (<>) nego se eksplicitno morao navesti tip koji se koristi u slučaju implementiranja apstraktnih klasa. Prilikom pokušaja da se tako nešto izvede kompajler prepoznaje grešku i ne dozvoljava pokretanje aplikacije. Sa novom verzijom aplikacije ovakva vrsta implementacije apstraktnih klasa je omogućena.

5. JAVA 10

Deseta verzija, isto kao i deveta, ne predstavlja verziju koja će imati dugoročnu podršku, ali već naredna verzija trebala bi da ima podršku sve do 2023. godine.

5.1 Paralelni *garbage collector* za G1

Deveta verzija donela je novi podrazumevani *garbage collector* G1 koji je dizajniran kako bi se smanjile pauze koje su pravljene sa prethodnim. Prilikom rada sa kolekcijama u paraleli dolazilo je do problema kada se memorija nije oslobađala dovoljno brzo i u tim trenucima bi imali situaciju kao u starijim verzijama. U novoj verziji Jave algoritam *full garbage collector-a* je paralelizovan i sada koristi isti broj niti kao i kolekcije. Kako G1 deli memoriju u jednake regione, ova promena rezultuje time da se manje prostora gubi i bolje se iskorišćava.

5.2 Definisanje promenljivih sa var tipom

Var je nova ključna reč u Java programskom jeziku koja omogućava deklarisanje promenljivih bez prethodnog navođenja tipa i predstavlja potpuno novi koncept. Lokalna promenljiva dobija tip na osnovu rezultata vrednosti koja mu je dodeljena, ako se istoj promenljivoj kasnije dodeljuje vrednost koja je različitog tipa dolazi do greške zbog loše konverzije između tipova. Prilikom deklarisanja promenljive sa *var* vrednost se mora odmah dodeliti inače kompajler vraća grešku. Ako se var promenljiva instancira sa operatorom dijamant, a da se pri tome ne navede tip neće biti moguće da se koriste metode koje su specifične za neki tip podataka iz razloga što će kompajler gledati na elemente kao *Object* tipove, ako se prilikom instanciranja navede tip sve metode će biti dostupne.

5.3 Dodavanja klasa u arhivu

Još u verziji 5 omogućeno je skladištenje klasa u arhivu i njihovo ponovno korišćenje kada je to potrebno. Glavna ideja je bila da se prilikom prvog pokretanja Java Virtualne Mašine sve klase koje se često koriste kompajliraju i skladište u arhivu na disku. Skladištenje je bilo moguće samo za klase koje su imale poverenje Jave, a nije bilo moguće dodati sopstvene klase. Od nove verzije dodata je mogućnost dodavanja sopstvenih klasa u arhivu i tom prilikom je otvorena mogućnost da se prilikom pokretanja aplikacija smanji potrebno vreme za kompajliranje.

5.4 Garbage collector interfejs

Još jedan interesantan dodatak novoj Javi koji povećava izolaciju koda za *garbage collector*-e, ali predstavlja čistiji interfejs koji je dostupan. U prethodnim verzijama moralo se objediniti znanje iz svih delova kako bi se određeni *garbage collector* izbacio iz upotrebe ili dodao novi. Novom verzijom Jave implementiranje *garbage collector-a* svelo se na implementiranje određenih metoda.

5.5 Dodavanje klasa u arhivu

Još u verziji 5 omogućeno je skladištenje klasa u arhivu i njihovo ponovno korišćenje kada je to potrebno. Glavna ideja je bila da se prilikom prvog pokretanja Java Virtualne Mašine sve klase koje se često koriste kompajliraju i skladište u arhivu na disku. Skladištenje je bilo moguće samo za klase koje su imale poverenje Jave, a nije bilo moguće dodati sopstvene klase. Od nove verzije dodata je mogućnost dodavanja sopstvenih klasa u arhivu i tom prilikom je otvorena mogućnost da se prilikom pokretanja aplikacija smanji potrebno vreme za kompajliranje.

5.6 Objedinjavanje JDK repozitorijuma u jedan

Većina novina 10. verzije Jave vezana je za sređivanje repozitorijuma, unapređenje delova koda koji je vezan za Java Virtuelnu Mašinu i generalno delova koji nisu lako dostupni krajnjem korisniku i omogućavaju pravilno izvršavanje koda koje je korisnik napisao. U prethodnoj verziji JDK (Java Development Kit) je bio podeljen u devet repozitorijuma: *root, corba, hotspot, jaxp, jaxws, jdk, langtools i nashorn*. Da se ne bi događali ovakvi problemi inženjeri su objedinili sve repozitorijume u jedan i tako omogućili lakše održavanje brže otklanjanje problema koji nastaju na Java Virtuelnoj Mašini.

5.7 Eksperimentalni Graal JIT (Just in time) kompajler

U novoj verziji postoji mogućnost da se omogući korišćenje JIT kompajlera, a njegova upotreba moguća je na Linux/x64 platformi i u zavisnosti od povratih informacija koje tim inženjera bude dobijao od korisnika postoji mogućnost da u budućnosti on postane standardan. Ovaj kompajler je deo JVM (Java Virtual Machine) kompajlera i njegov zadatak je da pomogne prilikom prevođenja bytecode-a u mašinski jezik, a sve to kako bi se omogućilo brže izvršavanje aplikacije pošto je izvršavanje bytecode-a dosta sporije od izvršavanja mašinskog.

6. ZAKLJUČAK

Vlasnici Java programskog jezika odlučili su da na svakih šest meseci predstavljaju novu verziju kako bi jezik ostao konkurentan i pre svega zanimljiv korisnicima, a i kako bi se eventualni nedostaci popunili.

Sa verzijama koje su opisane u ovom radu svakako zainteresovali okruženje i pružili dosta dobrih stvari koja do sada nisu bile prisutne. Pored novina i unapređenja koja su nove verzije donele dosta je rađeno i na bezbednosti tako da je Java jedan od najbezbednijih i najboljih jezika za pravljenje aplikacija, a to dokazuje i procenat aplikacija koje su napravljene sa njom.

7. LITERATURA

[1]https://en.wikipedia.org/wiki/Java_(programming_language) – Java programski jezik

[2] https://en.wikipedia.org/wiki/C%2B%2B – C++

[3] https://en.wikipedia.org/wiki/Java_bytecode - Bytecode

[4]https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html - Lambda izrazi
[5]https://docs.oracle.com/javase/tutorial/java/IandI/defaultmethods.html - Default-ne metode

[6] https://en.wikipedia.org/wiki/JavaScript - JavaScript
[7]https://docs.oracle.com/javase/10/nashorn/nashorn-java-api.htm#JSNUG119 – Nashron

[8]https://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html - Stream
[9]https://en.wikipedia.org/wiki/Asynchronous_method_invocation - Asinhroni pozivi

[10] https://en.wikipedia.org/wiki/Base64 - Base64

[11]https://docs.oracle.com/javase/9/gctuning/garbage-first-garbage-collector.htm#JSGCT-GUID-0394E76A-1A8F-425E-A0D0-B48A3DC82B42 – G1 garbage collector
[12] http://openjdk.java.net/jeps/11 - Incubator module

[13]https://www.tutorialspoint.com/design_pattern/builder_pattern.htm - Build pattern

Kratka biografija:

Dejan Urošević - rođen 1993. godine u Rumi. Završio je srednju gimnaziju „Branko Radičević“ u Staroj Pazovi 2012. godine. Osnovne akademske studije studijskog programa Računarstvo i automatika upisao je 2012. godine na Fakultetu tehničkih nauka, Univerziteta u Novom Sadu. 2016. godine završava osnovne studije. Master akademske studije na studijskom programu Računarstvo i automatika, Dejan upisuje te iste godine (2016.), pri čemu bira modul Elektronsko poslovanje. Dejan je ispunio sve obaveze i položio sve ispite predviđene studijskim programom.