ucenje Jave :) :) :)

---

nedoumice

--->

Zasto se koristi JVM i koja je tacno njena prednost i generalno prednost Java programa u odnosu na npr. C programe?

JVM sluzi da prevodi bytecode u masinske instrukcije platforme na kojoj se izvrsava java program.

Prednost je sto Java program radi na svim platformama bez ponovnog kompajliranja.

Sa druge strane, C programi se oslanjaju prvenstveno na clib (C biblioteke) koje se razlikuju od platforme do platforme.

Gdje koristimo Java Applet-e? Nigdje.

Kad koristimo metodu finalize()? (Primjer sa prezentacija)

Koja je poenta klase Class?

Gdje se koriste final static varijable i kako funkcionisu?

Razlika izmedju static i normal import?

Jar files?

Kljucna rijec final ne moze uticati na to da stanje objekta koji referencira final referenca ne bude izmijenjeno. ok?

Kompajler ce prijaviti situaciju u kojoj naredni catch blok ima za argument ciji je tip u hijerarhiji iznad tipa prethodnog catch bloka.

Navodno, naredni catch blok tada ne bi mogao da se izvrsi. Zasto?

Provjereni i neprovjereni (checked&unchecked) izuzeci?

Zasto naslijedjena static metoda ne smije biti redefinisana?

Nisam sigurna da li je ovo tacno: Kada redefinisemo neku metodu, to se desava u toku runtime-a, a staticke metode su vezane za klasu,

a ne nuzno za instancu.

Zasto naslijedjena static/instance metoda ne moze biti redefinisana kao instance/static metoda?

Dynamic bindings.

this.string1 = string1; //shallow copy?

Sta je runtime klasa tekuceg objekta?

Sta predstavlja npr. File datoteka = new File("f:\\test\\datoteka.txt"); kad nemam navedene direktorijume?

---

korisne stvari

---->

Postoje razlicite arhitekture procesora kao npr. x86, ARM.

Kratko poredjenje dva navedena procesora:

ARM is a RISC (Reduced Instruction Set Computing) architecture while x86 being a CISC (Complex Instruction Set Computing) one.

The core difference between those in this aspect is ARM instructions operates only on registers with a few instructions for

loading and saving data from/to memory while x86 can operate on directly memory as well.

Up until v8 ARM was a native 32 bit architecture, favoring four byte operations over others.

JRE: Java Runtime Environment. It is basically the Java Virtual Machine where your Java programs run on.

JDK: It's the full featured Software Development Kit for Java, including JRE, and the compilers and tools (like JavaDoc, and Java Debugger) to create and compile programs.

Primitivni tipovi podataka se cuvaju na steku, a objekti se cuvaju na heap-u (moze se pristupiti samo referencama objekata koje se nalaze na steku - nikad samim objektima).

In a multi-threaded application, each thread will have its own stack.

But, all the different threads will share the heap.

Because the different threads share the heap in a multi-threaded application, this also means that there has to be some coordination between

the threads so that they don’t try to access and manipulate the same piece(s) of memory in the heap at the same time.

Java permits primitive types in generics.

Autoboxing - when you pass byte to a generic method that expects Byte, the compiler automatically Autoboxes it to Byte which is an instance of Object.

The antithesis of that operation is called Auto Unboxing and that is why operations like the one shown below are legal.

int a = new Integer(5);

Integer b = 5;

Podaci tipa int se mogu pisati sa donjom crtom ("_") radi lakseg uocavanja.

Tacka (".") je separator (a ne operator)!

Samo char je neoznacen tip.

Default vrijednost za boolean podatak je false.

Unarni plus + promocija byte, short i char u int.

The unary plus operator performs an automatic conversion to int when the type of its operand is byte, char, or short.

This is called unary numeric promotion, and it enables you to do things like the following:

char c = 'c';

int i = +c;

Logicki operatori &&,|| i ! se mogu primjenjivati samo na boolean vrijednosti!

Sto se tice bitskih operatora, nad podacima tipa boolean mogu se primijeniti svi osim operatora logicke negacije nad bitovima (~).

Switch radi sa byte, short i int primitivnim tipovima (kao i sa Byte,Short i Int tipovima), sa enum tipovima i klasom String.

Za ostale slucajeve koristimo if else.

Jedini zadatak podrazumijevanog konstruktora jeste taj da pozove konstruktor klase iznad pozivom metode super();

Reference na nizove se čuvaju na steku, dok se elementi nizova čuvaju na heap-u.

Klasa ne moze biti istovremeno deklarisana kao final i abstract!

Anything that is a subclass of Exception except for RuntimeException/Error and its subclasses is a checked exception.

Deklaracija niza:

int[] niz; //ili

int niz[];

int matrix[][]; //ili

int[][] matrix; //ili

int[] matrix[];

Deklaracijom niza se ne kreira niz, već se samo deklariše referenca koja može referencirati objekat niza.

Optional objekti.

The purpose of Optional is not to replace every single null reference in your codebase but rather to help design better APIs in which—just by reading the signature of a method—users can tell whether to expect an optional value. In addition, Optional forces you to actively unwrap an Optional to deal with the absence of a value; as a result, you protect your code against unintended null pointer exceptions.

Privatni, redefinisani i sakriveni članovi osnovne klase nisu nasljeđeni.

Ako je izvedena klasa u istom paketu kao i roditeljska, nasljeđuje i friendly članove.

Clanovi koji imaju podrazumijevanu dostupnost u osnovnoj klasi ne mogu biti nasljeđeni od strane klasa u drugim paketima.

Exceptions rule in Inheritance goes like this:

"When a subclass overrides a method in super class then subclass method definition can only specify all or subset

of exceptions classes in the throws clause of the parent class method(or overridden method)".

Ako klasa implementira samo NEKE od metoda interfejsa, tada je ona apstraktna.

Jedan interfejs moze naslijediti vise interfejsa, i to kljucnom rijecju extend.

Enum ne moze naslijediti neku drugu klasu jer automatski vec nasljedjuje jednu - java.lang.Enum.

Klase Integer, Long, Float i Short posjeduju još jednu preklopljenu valueOf metodu, koja kao argument prima String i bazu brojnog sistema.

Kao ulazni i izlazni tokovi mogu se posmatrati sljedeći entiteti: nizovi bajtova ili karaktera, datoteke, cijevi (eng. pipe),

konzola ili mrežne konekcije.

Klase InputStream i OutputStream su dizajnirane za bajt tokove, dok su klase Reader i Writer dizajnirane za karakter tokove.

An InputStream reads bytes, and a Reader reads characters.

All implementations of DataInput methods use EOFException instead of return values.

The correct type to use for currency values is java.math.BigDecimal.

InputStreamReader converts an InputStream to a Reader, and OutputStreamWriter converts an OutputStream to a Writer.

File.separator;

List<Animal> animals = new ArrayList<Animal>();

Class FileOutputStream:

FileOutputStream is meant for writing streams of raw bytes such as image data. For writing streams of characters, consider using FileWriter.

Class DataOutputStream:

A data output stream lets an application write primitive Java data types to an output stream in a portable way.

An application can then use a data input stream to read the data back in.

Preferences API is designed to store and retrieve user preferences and program-configuration settings.

In Java, it appears that you are supposed to use a File object to determine whether a file exists, because if you open it as a FileOutputStream

or FileWriter, it will always get overwritten.

ByteBuffer - ne postoje put i get metode za rad sa objektima, čak ni za rad sa stringovima.

Metode koje se bave citanjem/pisanjem na stek/u bafer moraju biti sinhronizovane.

public final void notify()

This method should only be called by a thread that is the owner of this object's monitor.

A thread becomes the owner of the object's monitor in one of three ways:

1. By executing a synchronized instance method of that object.

2. By executing the body of a synchronized statement that synchronizes on the object.

3. For objects of type Class, by executing a synchronized static method of that class.

Iterator se koristi umjesto for-each konstrukcije kada je neophodno:

–ukloniti tekući element

for-each konstrukcija sakriva iterator, tako da se ne može pozvati remove()

–iterirati preko višestrukih kolekcija paralelno

Preporuka je da se NavigableSet koristi umjesto SortedSet interfejsa

HashSet ima bolje performanse od TreeSet-a kada je u pitanju pretrazivanje, dok TreeSet ima bolje performanse od

HashSet-a kada je u pitanju ubacivanje i pretrazivanje elemenata sortiranog skupa.

So choice of usage depends entirely on your needs but I feel that even if you need an ordered collection, then you should still prefer

HashSet to create the Set and then convert it into TreeSet - e.g. SortedSet<String> s = new TreeSet<String>(hashSet);

Uobicajene implementacije pojedinih kolekcija:

Set interfejs - HashSet

List interfejs - ArrayList

Map interfejs - HashMap

Queue interfejs - LinkedList

Generički tip koji nije deklarisan kao final može biti nasljeđen.

Moguće je da generički tip naslijedi negenerički.

Moguće je i da konkretan tip naslijedi parametrizovani tip.

Konkretna klasa ne može naslijediti generički tip.

U slucaju datoteke .java sa vise definicija klasa:

-> svaka definicija klase iz iste izvorne datoteke kompajlira se u zasebnu datoteku sa .class ekstenzijom;

-> samo jedna klasa od svih u navedenoj datoteci moze biti public - ostale moraju biti private

Objekti nemaju isti opseg vidljivosti kao primitivni tipovi i reference,

jer objekat će postojati na heap-u sve dok ne bude uklonjen od strane garbage collector-a!

Ako je modifikator pristupa tipa podrazumijevani, to znaci da je tip dostupan najvise do nivoa paketa u kojem se nalazi.

Napomena: Apstraktna klasa ne moze biti instancirana.

Metoda klase ne moze biti apstraktna i privatna, jer onda ta metoda (koja mora biti override-ovana u podklasi, nece joj biti dostupna).

Ako je klasa apstraktna, ne mora, a moze imati apstraktne metode. Medjutim, ako u klasi postoji bar jedna apstraktna metoda,

klasa mora biti oznacena kao apstraktna!

Ako neka klasa naslijedi apstraktnu klasu, potrebno je da implementira sve njene apstraktne klase; ukoliko se to ne sprovede,

tada je i klasa nasljednica apstraktna klasa!

Interfejsi specifikuju samo apstraktne metode.

Varijabla sa modifikatorom pristupa private ne moze postojati u interfejsu, jer nema smisla proglasavati je privatnom kad znamo da je nikada

necemo koristiti (SVE metode interfejsa su apstraktne).

In Java, an abstract class can implement an interface, and not provide implementations of all of the interface's methods.

It is the responsibility of the first concrete class that has that abstract class as an ancestor to implement

all of the methods in the interface.

String hello = null; hello.valueOf(123); works!

No, we can't have abstract enums, all enum types extend Enum, and they're final by default.

Since Java does not support multiple inheritance, an enum cannot extend anything else.

However an enum can implement Interface.

Yes, you can define abstract methods in an enum declaration if and only if:

all enum values have custom class bodies

with implementations of those methods.

Example:

```
public enum Foo {
  BAR {
    public void frobnicate() {
      // do BAR stuff
    }
  },
  BAZ {
    public void frobnicate() {
      // do BAZ stuff
    }
  };

  public abstract void frobnicate();
}
```

Modifikator final se koristi u deklaraciji klase kako bi označio da klasa ne može biti nasljeđena.

Bitno! Naslijedjivanje statickih clanova:

the only difference with inherited static (class) methods and inherited non-static (instance) methods is that

when you write a new static method with the same signature, the old static method is just hidden, not overridden.

Static methods can not be overridden in the exact sense of the word, but they can hide parent static methods.

Staticke metode ne mogu biti redefinisane, prema tome ne mogu biti oznacene kao apstraktne!

There are really three important subcategories of Throwable:

Error - Something severe enough has gone wrong the most applications should crash rather than try to handle the problem,

Unchecked Exception (aka RuntimeException) - Very often a programming error such as a NullPointerException or an illegal argument.

Applications can sometimes handle or recover from this Throwable category -- or at least catch it at the Thread's run() method,

log the complaint, and continue running.

Checked Exception (aka Everything else) - Applications are expected to be able to catch and meaningfully do something with the rest,

such as FileNotFoundException and TimeoutException...

Catch blok ne mora postojati,ali mora blok finally mora.

Catch ili finally se ne mogu pojaviti prije try bloka i ne mogu se pojaviti bez try bloka!

The default access modifier (if not private, protected or public is specified) is package private,

which means that it may be accessed from any other class in the same package as the declaring class.

**********Access Levels**********

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
default	Y	Y	N	N
private	Y	N	N	N

****

Jedna klasa se ne moze nalaziti u vise paketa!

Pri redefinisanju (override) metoda, sve treba biti identicno (dostupnost, naziv, povratni tip, redoslijed i tip argumenata, naziv),

sa bitnom napomenom da povratni tip moze biti tacan tip ili podtip tog tipa; isto vazi za izuzetke, kao i za dostupnost.

Nije dovoljno da se metode razlikuju samo po povratnom tipu da bi bile preklopljene, potrebno je da se razlikuju i po

listi parametara.

Kod preklapanja metoda nema ogranicenja za povratni tip, izuzetke i dostupnost.

Polje u klasi nasljednici moze maskirati staticko polje osnovne klase - tip polja nije bitan, bitan je samo naziv.

Preklapanje - samo isti naziv.

Redefinisanje - izvrsava se metoda tipa objekta koji je referenciran za vrijeme runtime-a.

Preklapanje - izvrsava se metoda tipa objekta cijom je referencom deklarisan.

In the Java programming language, a method signature is the method name and the number and type of its parameters.

Return types and thrown exceptions are not considered to be a part of the method signature.

Sve metode interfejsa su implicitno abstract I public, a sve konstante u interfejsu su public static final.

Konstante u interfejsima moraju biti inicijalizovane (has to to with final).

Kompajler ce javiti gresku ako na listi polja klase stoji npr.:

private static final String foo;

Metoda toString vraca NazivKlase@hash_vrijednost_objekta

Klasa StringBuffer je sinhronizovana.

Instance klase FILE su nepromjenljive.

Static Variables are not serialized, so during deserialization static variable value will loaded from the class.

(Current value will be loaded.)

Java programi razlikuju 2 vrste niti: korisničke niti i demonske niti.

Java program završava svoje izvršavanje kada se završe sve njegove korisničke niti, bez obzira na to

da li se trenutno izvršava jedna ili više demonskih niti.

Allowing a thread to acquire the monitor it already owns is called Reentrant Synchronization

and without which it'll be very difficult to ensure that a thread in Java doesn't block itself.

Promjenljive definisane unutar interfejsa su implicitno final promjenljive.

Metoda ne moze biti:

- staticka i apstraktna;
- final i apstraktna.
- private i apstraktna;

Promjenjiva ne moze biti synchronized! Samo metode mogu biti synchronized!

Metoda ne moze biti transient! Samo promjenjiva moze biti transient!

Gore navedeno isto vazi za volatile.

“… the volatile modifier guarantees that any thread that reads a field will see the most recently written value.” - Josh Bloch

Nakon završetka catch bloka, izvršavanje se nastavlja u finally bloku, ako je ovaj blok specificiran,

i to bez obzira na to da li je novi izuzetak bačen u catch bloku.

Situacija u kojoj catch blok ima argument čiji je tip u hijerarhiji iznad tipa argumenta nekog od narednih catch blokova

nije dozvoljena – ovakvu situaciju prijaviće kompajler, jer naredni catch blok ne bi nikad mogao da se izvrši.

Ako je članu osnovne klase moguće pristupiti navođenjem naziva tog člana,

bez bilo kakve dodatne sintakse (poput ključne riječi super), onda se taj član smatra nasljeđenim.

Iz tog razloga privatni, redefinisani i sakriveni članovi osnovne klase nisu nasljeđeni.

Redefinisani? Nije tacno? npr. toString?

Ukoliko u roditeljskoj klasi definisemo samo konstruktor sa parametrima, potrebno je u naslijedjenoj klasi pozivati bas taj konstruktor;

pozivanje default konstruktora dovesce do greske pri kompajliranju.

Interfejs ne moze imati privatne clanove.

Kompajler tjera programera da inicijalizuje promjenjivu u interfejsu, kao i da postavi vidljivost na public.

There are several differences between HashMap and Hashtable in Java:

1. Hashtable is synchronized, whereas HashMap is not.

   This makes HashMap better for non-threaded applications.

2. Hashtable does not allow null keys or values.

   HashMap allows one null key and any number of null values.

3. One of HashMap's subclasses is LinkedHashMap, so in the event that you'd want predictable iteration order

   (which is insertion order by default), you could easily swap out the HashMap for a LinkedHashMap.

   This wouldn't be as easy if you were using Hashtable.

Unutrasnja neimenovana klasa moze naslediti tacno jednu klasu ili implementirati tacno jedan interfejs.

Metoda lokalne unutrasnje klase moze biti apstraktna.

Apstraktna klasa moze imati konstruktor - koristi se u naslijedjenim klasama koristenjem kljucne rijeci super.

Apstraktna klasa moze imati main.

Enum kolekcija moze imati konstruktor i varijable.

Ako pristupamo iz unutrasnje klase lokalnoj promjenljivoj, ona mora biti final.

Velicine primitivnih tipova:

byte: 8-bit

char: 16-bit

short: 16-bit

int: 32-bit

long: 64-bit

float: 32-bit

double: 64-bit

19 specific conversions on primitive types are called the widening primitive conversions:

->byte to short, int, long, float, or double

->short to int, long, float, or double

->char to int, long, float, or double

->int to long, float, or double

->long to float or double

->float to double

Everything else needs an explicit cast.

wait,notify I notifyAll su metode Object klase.

Metoda instance moze biti redefinisana u klasi nasljednici u apstraktnu metodu vaze isti uslovi kao za redefinisanje u obicnu metodu instance.

You should always close the output and input stream before you close the socket.

int[] brojeviPitanja = new Random().ints(1, 5).distinct().limit(3).toArray();

java -classpath "C:\Users\HP KORISNIK\Desktop\Workspace" klijentserver.kviz.Server

http://www.oracle.com/technetwork/articles/javase/rmi-corba-136641.html

Primjer 1:

```
Integer n1 = new Integer(45);
Integer n2 = new Integer(45);
System.out.println(n1==n2); //false
System.out.println(n1.equals(n2)); //true
```

Primjer 2:

```
if(test1() && test2() && test3());//ako npr. test1() bude false, test2() i test3() se nece ni pozvati!
```

Primjer 3:

```
String s1 = "Ovo je string!";
String s2 = "Ovo je string!";
String s3 = "Ovo je" + " " + "string!";

String temp = "Ovo je";
String s4 = temp + " string!"; //s4 nece dijeliti isti objekat sa s1,s2 i s3 jer se ne izracunava za vrijeme kompajliranja
```

Preskocila sam (ali se moram vratiti na to):

lab8 (2. zadatak)