**P1 izpit (**[**https://docs.oracle.com/en/java/javase/21/docs//api/java.base/java/lang/**](https://docs.oracle.com/en/java/javase/21/docs//api/java.base/java/lang/)**)**

**TEST VHOD-IZHOD:** tj.exe <Program.java> . . **oz:** tj.exe <Program.java> <Mtesti> <Mrezultati>

**TEST RAZRED-IZHOD:** tj.exe -t 1000 -times -p 6-9 -threads 3 **oz:** tj.exe <Mrazredi> <Mtesti> <Mrezultati>

**ROČNO TESTIRANJE (javac):** java Program < test07.in > test07.res **nato pa:** fc test07.res test07.out **oz.** echo 5 6 | java Vsota **(**start prikaz.htm**)**

* (int) ‘A’ = 65, (int) ‘a’ = 97, (int) ’0’ = 48
*  (a % b ima isti predznak kot a)
* “a” + st1 + st2 vrne niz: “a[st1][st2] in ne vsote!
* Switch: byte, short, int, char, String
* Random r = new Random(long seme); int r.nextInt(n) // vrne [0, n) **ali** double Math.random() // vrne [0, 1)
* A screen shot of a computer code

  Description automatically generatedSystem.out.printf(“%5d %-3f %c %s %b %5.2f %06d”, ...); // d bo zasedal skupaj najmanj 5 mest (rezervira); f bo poravnan na levo, vsaj 3 mesta; drugi f bo izpisan na dve decimalni mesti; drugi d bo imel spredaj toliko ničel, da bo zasedal najmanj 6 mest
* long Math.round(double a); // vrne pravilno zaokroženo št.

public static <T> void metoda(String a);

public class A extends B implements Vmesnik1, Vmesnik2{

* če razred implementira Iterable ali če je array: for(T element : t) {}
* namesto setterjev pogosto raje naredimo in vrnemo nov objekt
* Statični atributi enaki za vse objekte razreda – ponavadi inicializiramo takoj ob deklaraciji ali z inicializatorjem “ ”.repeat(5);
* Klic konstruktorja razreda: this(a, b);, klic konstruktorja nadrazreda: super(a, b); - prva vrstica

ce rabimo rezultat metode, pa je ta implementirana šele v podrazredih, naredimo abstraktno metodo in jo lahko uporabimo

* Klic metode nadrazreda: super.metoda()
* private static final int A = 10; // statična konstanta
* @Override – redefiniranje metod nadrazredov/vmesnikov
* Podrazred podeduje vse neprivatne lastnosti in metode
* A computer code with numbers and symbols

  Description automatically generatedobjekt *instanceof* Razred // vrne true če je objekt tipa Razred ali njegovega podtipa; omogoča nam castanje iz nadtipov v podtipe brez izjem (if stavek)
* tabela lahko hrani objekte deklariranega tipa ali njegovih podtipov
* double Math.pow(double a, double b)
* double Math.sqrt(double a)
* Tip.MIN\_VALUE

Isto za continue

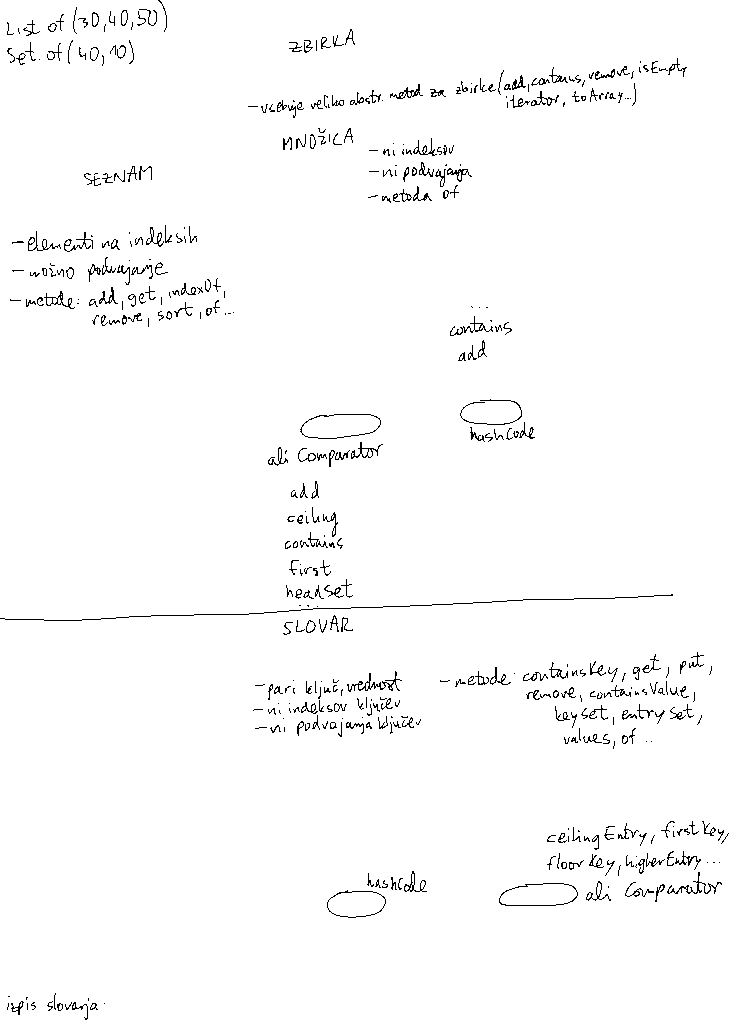
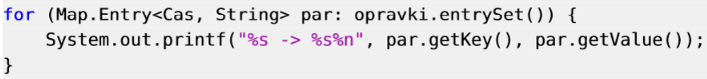
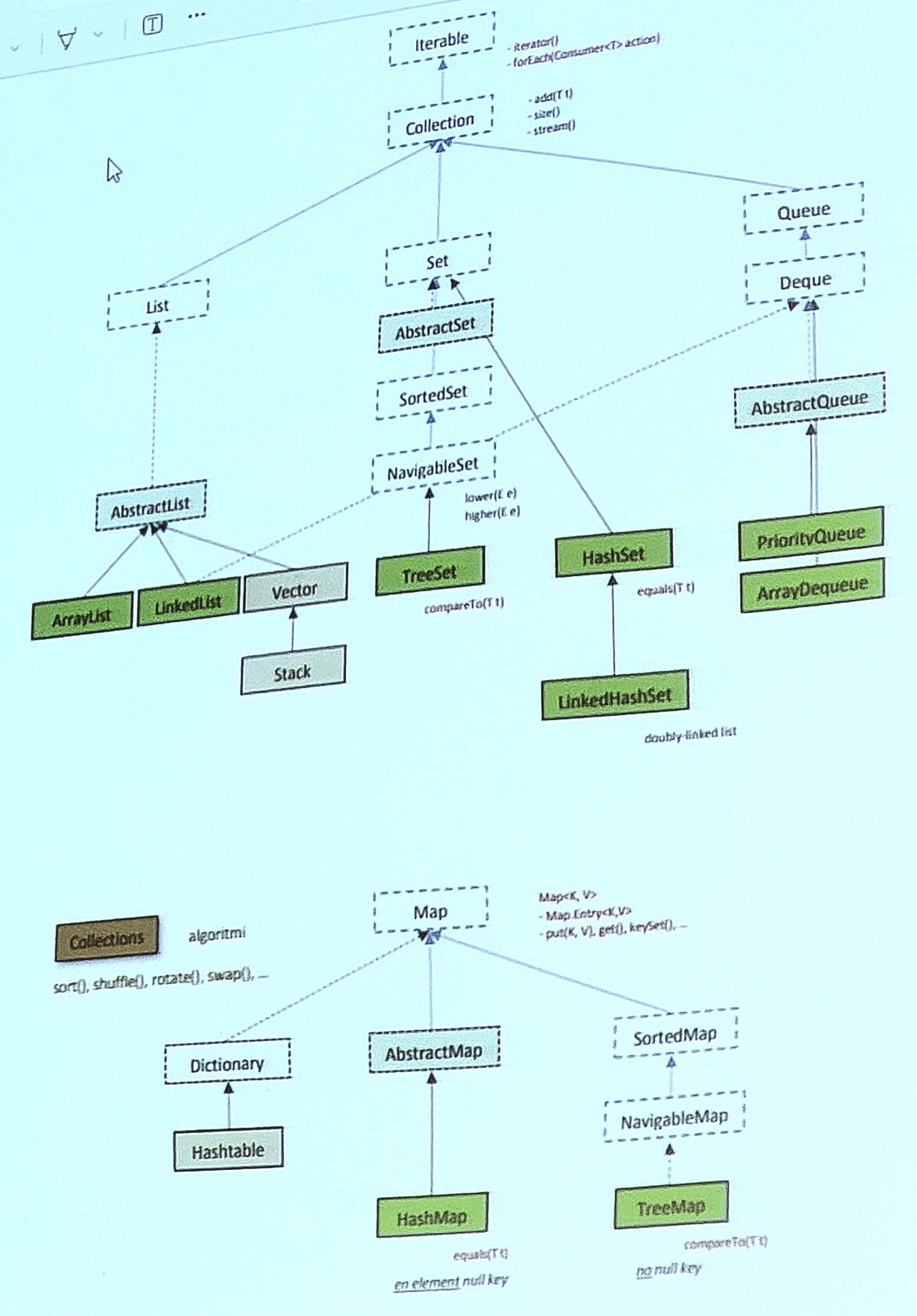
* int Integer.parseInt(...) **ali** Integer Integer.valueOf(...)
* String String.valueOf(...) **ali** String Integer.toString(...)
* A black text on a white background

  Description automatically generatedScanner.hasNextInt() **ali** Scanner.hasNext()
* Scanner.nextLine()
* char[] String.toCharArray()
* **IZPIS VEČDIM. TABEL:** Arrays.deepToString() vrne niz
* **KOPIRANJE TABEL:** Arrays.copyOf() **samo za enodim. tabele,** tab.clone()?
* razred Object vsebuje metode (ki jih lahko @Override v vseh podrazredih):
  + public String toString(){ return String.format(“...”, ...) ;} // Sysout(razred);
  + public boolean equals(Object obj);
  + public int hashCode();
* **EQUALS:**
* (z == primerjamo reference, equals po defaultu enako, lahko ga redefiniramo), (**PRI String JE ŽE OVERRIDAN IN VRNE true TUDI ČE IMATA ENAKI VSEBINI**); **ČE EQUALS VRNE TRUE, MORA TUDI int hashCode() vrniti enako za oba objekta**

@Override public boolean equals(Object obj){ if(this == obj) return true; if(!(obj instanceof Cas)) return false;

Cas objCas = (Cas)obj; // potrebno castanje (Object lahko castamo v karkoli - po prevajalniku, izvajalnik pa lahko sproži izjemo) return (this.ura == objCas.ura && this.min == objCas.min);}

* **HASHCODE:** naj bi veljalo: 



* Vrne celo št. ki predstavlja identiteto objekta, ni nujno da za vsaka različna vrne različno, čeprav večinoma da

A black text on a white background

Description automatically generated A computer code with black text

Description automatically generated

**Ovojni tipi: referenčni tipi za primitivne tipe**

* int st = 0**;** Integer a = Integer.valueOf(st)**;** int st2 = a.intValue(); // ni potrebno
* če je vrednost ovojnega tipa null, lahko dobimo nullPointerException

**Generiki/parametrizirani tipi: ZGOLJ ZA REFERENČNE TIPE**

* public class Ovojnik<T extends Number, ? super U, V>{ private T a; public Ovojnik(T a) {...}; ... } **/\* ? ... poljuben referenčni tip \*/ in** Ovojnik<String> p = new Ovojnik<>(“...”);

// T, U, V lahko uporabljamo znotraj razreda, vendar NE ZA STATIČNE METODE IN STATIČNE ATRIBUTE

* public static <T> boolean metoda(Ovojnik<T>){...} **in klic:** Razred.<String>metoda(o);
* tabele:

@SupressWarnings(“unchecked”)

T[] tab = (T[]) new R[10]; // R je tip, s katerim je T navzgor omejen (T extends R), ponavadi Object

**VMESNIKI:** abstraktni razred ki v osnovi vsebuje le abstraktne metode (statične metode, privzete metode, statične konstante);

* Razred implementira vmesnik – implementira vse njegove abstraktne metode
* Predstavlja neko možnost, kaj lahko storimo z razredom (public interface -ljiv/-able{})

***COMPARABLE*: int compareTo(T drugi) – NARAVNA UREJENOST: objekte razreda T naravno primerjamo med sabo (če this sodi pred drugi, vrne negativno)**

***COMPARATOR*: int compare(T prvi, T drugi) – ALTERNATIVNA UREJENOST za objekte razreda Tip (če prvi sodi pred drugi, vrne negativno), implementacija: 1. V razredu Tip (najbolj smiselno) private static class Primerjalnik implements Comparator<Tip>{...}, ali 2. anonim. notr. razred v metodi: static Comparator<Tip> poNecem(){ return new Comparator<Tip>() {...}; } ali 3. z lambdo v metodi: static Comparator<Tip> poNecem(){ return (a, b) -> ... } (PONAVADI IMAMO V RAZREDU Tip METODO, KI VRNE TA COMPARATOR)**

*\_\_Arrays.*sort(T[], Comparator<T>)\_\_

**ITERABLE: Iterator<T> iterator() – sprehodljiv razred vrne iterator, s katerim se lahko sprehodimo čez objekt oz. elemente objekta tega razreda T (običajno vsebovalnik, s foreach zanko)**

**ITERATOR:** **boolean hasNext() in T next() – kako se sprehodimo po objektu razreda, hasNext vrne throw new NoSuchElementException ali vrne naslednji element tipa T (LAHKO DEFINIRAMO VEČ ITERATORJEV) (z metodo Collection.iterator() lahko pridobimo iterator za zbirke)**

**VSEBOVALNIKI:** objekti za hranjenje poljubnega števila elementov referenčnih tipov (generični razredi)

* A screenshot of a computer code

  Description automatically generatedRazred Collections za delo z vsebovalniki, npr: Collections.max/min(imena), Collections.sort(imena), Collections.binarySearch(imena, “”), Collections.swap(imena, 1, 2), Collections.shuffle(imena, r) ......



**LAMBDE:** ***UPORABIMO, KO PODAJAMO FUNKCIJO (vmesnika) V DRUGO FUNKCIJO*** - skrčena izdelava objekta funkcijskega **vmesnika – VRNE KAZALEC NA OBJEKT RAZREDA, KI IMPLEMENTIRA VMESNIK**

* Nekateri funkcijski vmesniki: IntBinaryOperator
*  A close up of text

  Description automatically generated

Možno tudi z vmesnikom Predicate.

* A screenshot of a computer code

  Description automatically generated potrebujemo vmesnik z metodo void in parametrom T t, tak je vmesnik Consumer z metodo accept:

**REKURZIJA:**

* Potreben je ustavitveni pogoj
* Vračanje vrednosti – rekurzivno drevo
* Da se izognemo večkratnemu računanju pri istih vrednostih, uporabimo tabelo za memoizacijo, ki jo podamo kot parameter rekurzivne funkcije in kličemo če je že izračunana, izračunano vrednost moramo posebej shraniti v tabelo memo preden jo vrnemo

A computer screen shot of a computer code

Description automatically generated

A diagram of a network

Description automatically generated

* **Praštevila**:

Boolean[] z n+1 elementi // ce je true, je sestavljeno

Meja = (int) Math.round(Math.sqrt(n))

For [2:meja] if !array[i]: For [2\*i:n s korakom p] array[j] = true

* **GCD**:

A table with black text

Description automatically generatedpublic int gcd(int a, int b) {

if (b==0) return a;

return gcd(b,a%b);

}