Leksioni 5
The object class

5.1 Cloning
5.2 Strings
5.3 The JDK LinkedList class

5.1 Klonimi në Java

Klonimi në Java është një temë e rëndësishme për programuesit dhe për zhvilluesit e aplikacioneve. Me përdorimin e klonimit, ne mund të krijojmë një kopje të saktë të një Objekti. Klonimi krijon një instancë të re të klasës së objektit aktual dhe e inicializon atë. Gjithmonë kjo realizohet duke krijuar një objekt të ri (objekt i klonuar) me saktësisht përmbajtjen e objektit aktual.
Le të themi që celulari është objekt dhe ne duam ta klonojmë atë. Do të thotë që ne duam të krijojmë një kopje të saktë të celularit.
Në këtë leksion, ne do të diskutojmë si mund ta arrijmë klonimin duke përdorur metodën e clone() dhe çfarë është metoda e klonimit në java. Shumica prej nesh tashmë e di cdo të thotë klonim, por në këtë leksion, ne do të diskutojmë disa gjëra unike në lidhje me klonimin.

5.1.1 Si ndertohet metoda e clone () në Java?

Metoda clone () përcaktohet në klasën e objektit e cila është klasa bazë (superklasa) në java. Kjo metodë përdoret për të krijuar një kopje të saktë të një objekti. Krijon një objekt të ri dhe kopjon të gjitha të dhënat e objektit të dhënë në atë objekt të ri.

Le të shohim kodin:

```
protected native Object clone() throws CloneNotSupportedException;
```

5.1.2 Çfarë është klonimi në Java?

Java ofron një mënyrë për të krijuar kopjen e saktë të Objektit. Le të themi se keni një objekt me veti të ndryshme. Tani dëshirojmë të krijojmë një objekt të ri me të njëjtat veti dhe vlera. Atëherë duhet të përdorim konceptin e klonimit të Java.
Disa programues e keqkuptojnë konceptin e klonimit me operatorin e barazimit (=). Java nuk siguron asnjë operator për të krijuar një kopje të një objekti.
Nëse përdorni operatorin e barazimit (=) atëherë ai do të krijojë një kopje të ndryshores referuese dhe jo të objektit.

```
Student object1 = new Student();
Student object2 = object1;
```

Me përdorimin e operatorit të caktimit objekt1 dhe objekt2 tregoni në të njëjtin vend, nëse ndonjë ndryshim në objekt2 do të reflektohet në objekt1.

Le ta diskutojmë me një shembull:

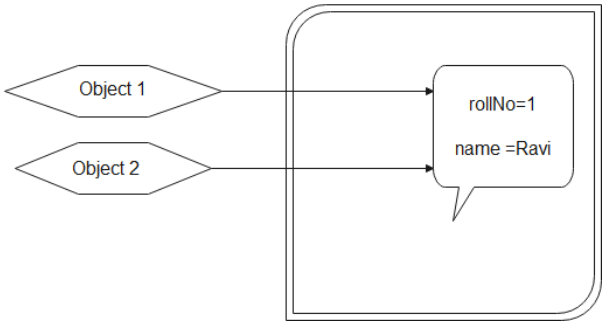
```
class Student
{
int rollNo;
String name;
}
public class Data
{
public static void main(String[] args)
{
Student object1 = new Student();
object1.name = "Ravi";
```

```
object1.rollNo = 1;
Student object2 = object1;
// Changing the value in object2
object2.name = "Ram";
object2.rollNo = 2;
System.out.println("Printing values from Object1:");
System.out.println("Name:"+object1.name + " RollNo:"+ object1.rollNo);
System.out.println("Printing values from Object2:");
System.out.println("Name:"+object2.name + " RollNo:"+ object2.rollNo);
}
}
```

Output: *Printing values from Object1:*

Name:Ram RollNo:2
Printing values from Object2:
Name:Ram RollNo:2

Siç mund ta shihni në shembullin e mësipërm, të dy referencat (objekti1 dhe objekti2) mbajnë të njëjtën vlerë sepse ato tregojnë për të njëjtin objekt.

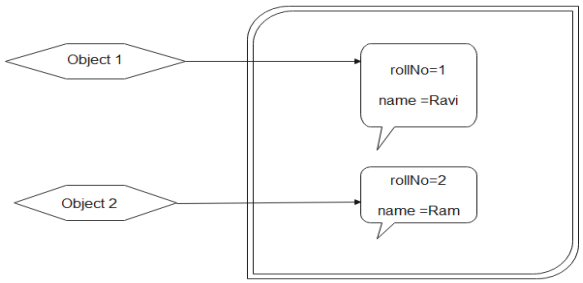


Kjo çështje zgjidhet me metodën clone () sepse krijon një objekt të ri dhe të dy objektet mbeten të pavarur.

```
class Student implements Cloneable
{
int rollNo;
String name;
public Object clone() throws CloneNotSupportedException
{
return super.clone();
}
}
public class Data
{
public static void main(String[] args) throws CloneNotSupportedException
{
Student object1 = new Student();
object1.name = "Ravi";
object1.rollNo = 1;
Student object2 = (Student) object1.clone();
System.out.println("Printing values from Object1:");
System.out.println("Name:"+object1.name + " RollNo:"+ object1.rollNo);
System.out.println("Printing values from Object2:");
System.out.println("Name:"+object2.name + " RollNo:"+ object2.rollNo);
// Changing the value in object2
object2.name = "Ram";
object2.rollNo = 2;
System.out.println("After changes in values");
System.out.println("Printing values from Object1:");
```

```
System.out.println("Name:"+object1.name + " RollNo:"+ object1.rollNo);
System.out.println("Printing values from Object2:");
System.out.println("Name:"+object2.name + " RollNo:"+ object2.rollNo);
}
}
```

Output: *Printing values from Object1: Name:Ravi RollNo:1*
Printing values from Object2:Name:Ravi RollNo:1
After changes in values
Printing values from Object1: Name:Ravi RollNo:1
Printing values from Object2: Name:Ram RollNo:2



Metoda clone () krijon kopje duke krijuar një objekt të ri dhe më pas duke kopjuar fushat.
SHENIM: Nëse klasa ka vetëm anëtarë primitivë të tipit të të dhënave, atëherë krijon një kopje të re të secilit anëtar të të dhënave të objektit. Por nëse klasa përmban një anëtar të tipit të të dhënave referuese gjithashtu atëherë nuk krijon një kopje të re të anëtarit të referencës (Thjesht krijon një kopje të re të primitivit). Referencat e anëtarëve si në objektin origjinal ashtu edhe në objektin e klonuar i referohen të njëjtit objekt. Në atë rast, si objekti origjinal ashtu edhe kopja e objektit (Objekti i klonuar) do të tregojnë për të njëjtin objekt në klaster. Kjo njihet si një kopje siperfaqësore.

5.1.3 Si të përdorim metodën clone ()?

1. Nëse dëshironi të përdorni klonimin në Java, duhet të anashkaloni metodën e clone () në klasën tuaj. Siç e dini tashmë metoda clone () ekziston në klasën e objektit, e cila ka përfituar në secilën klasë, ju mund ta anulloni atë.
2. Për të përdorur metodën clone (), duhet të implementojmë ndërfaqen Cloneable e cila ekziston është paketa java.lang. Shtë një ndërfaqe shënuese që përdoret për t'i thënë JVM se mund të kryejmë një klon në objektin tonë. Nëse nuk po përdorim ndërfaqen Cloneable, atëherë JVM do të hedhë CloneNotSupportedException kur do të thërrasim metodën clone ().
3. Kurdoherë që jemi duke anashkaluar metodën clone (), duhet të quajmë gjithashtu metodën clone () pjesë të superklasës. Do të bëjë një thirrje që është kloni i super (). Ky zinxhir do të vazhdojë derisa të arrijë metodën clone () të klasës Object.

Pika të rëndësishme në lidhje me klonimin:

1. Objekti i cili kthehet nga metoda clone () është një kopje e saktë e objektit origjinal. Objekti i kopjuar njihet si klon i objektit origjinal. Origjinali dhe kloni janë dy objekte të ndara në memorien. Deklarata më poshtë do të kthehet e gabuar.

```
// It returns false, because both are different object in heap memory
a.clone() == a;
```

Metoda klon () kthen një kopje të saktë të objektit origjinal.

```
// It returns true, because both object contains same values
x.clone().equals(x);
```

5.1.4 Si funksionon metoda clone ()?

Klasa Object siguron zbatimin e paracaktuar të metodës clone ().

Hapi 1: JVM kontrollon nëse klasa zbaton ndërfaqen Cloneable, e cila është një ndërfaqe shënjuese. Nëse klasa nuk implementon Cloneable Interface atëherë ajo hedh CloneNotSupportedException.

Hapi 2: JVM kontrollon përjashtimin e kontrolluar, i cili gjithmonë kërkohet të trajtohet gjatë klonimit të një objekti.

Hapi 3: Metoda Object.clone () krijon një kopje të cekët të objektit dhe ia kthen superklasës. Kopjon të gjithë përmbajtjen nga një objekt në një objekt tjetër.

5.1.5 Si funksionon JVM me klonimin e paracaktuar?

Pra, JVM kur thirret për klonim (klonimi i cekët), bën gjërat e mëposhtme:

1. Nëse klasa ka vetëm anëtarë të tipit primitiv të të dhënave, atëherë do të krijohet një kopje krejtësisht e re e objektit dhe referenca për kopjen e objektit të ri do të kthehet.

2. Nëse klasa përmban anëtarë të çfarëdo lloji të klasës, atëherë kopjohen vetëm referencat e objekteve ndaj atyre anëtarëve dhe kështu referencat e anëtarëve si në objektin origjinal ashtu edhe në objektin e klonuar i referohen të njëjtit objekt.

Llojet e klonimit Ekzistojnë dy lloje të klonimit.

1. Klonimi i cekët (sipërfaqësor)

2. Klonimi i thellë

1. Klonimi i cekët. Zbatimi i paracaktuar i siguruar nga Object.clone () po përdor një kopje të cekët. Në klonimin e cekët, metoda klon () krijon një objekt të ri dhe kopjon të gjitha fushat e objektit të klonueshëm në atë objekt të ri, duke krijur një kopje të re të fushës nëse është primitive. Nëse lloji i referencës në fushë, atëherë vetëm referencat e atij objekti do të klonohen në vend të krijimit të objektit të ri. Prandaj, ndryshorja referuese e të dy objekteve do të tregojë të njëjtin objekt. Le ta diskutojmë me shembull

```
class Address
{
int pinCode;
String city;
Address()
{ }
Address(int pinCode, String city)
{
this.pinCode = pinCode;
this.city = city;
}
}
class Student implements Cloneable
{
int rollNo;
String name;
Address address = new Address();
public Object clone() throws CloneNotSupportedException
{
return super.clone();
}
}
public class Data
{
public static void main(String[] args) throws CloneNotSupportedException
{
```

```
Student object1 = new Student();
object1.name = "Ravi";
object1.rollNo = 1;
object1.address = new Address(101, "CHD");
Student object2 = (Student) object1.clone();
System.out.println("Printing values from Object1:");
System.out.println("Name:"+object1.name + " RollNo:"+ object1.rollNo);
System.out.println("Address:"+  object1.address.pinCode  +  "  and  "  +
object1.address.city);
System.out.println("Printing values from Object2:");
System.out.println("Name:"+object2.name + " RollNo:"+ object2.rollNo);
System.out.println("Address:"+  object2.address.pinCode  +  "  and  "  +
object2.address.city);
System.out.println("Is both object are same in heap memory:" + (object1
== object2));
System.out.println("Is  both  Address  object  are  same  in  heap
memory:"+(object1.address == object2.address));
}
}
```

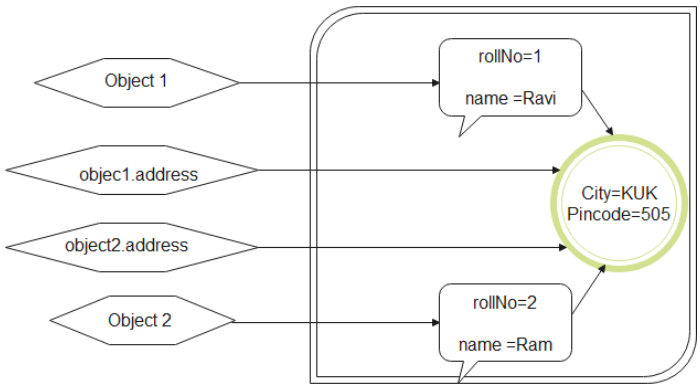
Output: *Printing values from Object1: Name:Ravi RollNo:1*
Address:101 and CHD
Printing values from Object2: Name:Ravi RollNo:1
Address:101 and CHD
Is both object are same in heap memory: false
Is both Address object are same in heap memory:true

Një ndryshim është bërë në objektin referent (Address) , atëherë ne objektin1 dhe objektin 2 do të pasqyrohet një object tjetër.

```
class Address
{
int pinCode;
String city;
Address()
{ }
Address(int pinCode, String city)
{
this.pinCode = pinCode;
this.city = city;
}
}
class Student implements Cloneable
{
int rollNo;
String name;
Address address = new Address();
public Object clone() throws CloneNotSupportedException
{
return super.clone();
}
}
public class Data
{
public static void main(String[] args) throws CloneNotSupportedException
{
Student object1 = new Student();
object1.name = "Ravi";
object1.rollNo = 1;
```

```
object1.address = new Address(101, "CHD");
Student object2 = (Student) object1.clone();
object2.name = "Ram";
object2.rollNo = 2;
object2.address.city = "KUK";
object2.address.pinCode = 505;
System.out.println("Printing values from Object1:");
System.out.println("Name:"+object1.name + " RollNo:"+ object1.rollNo);
System.out.println("Address:"+  object1.address.pinCode  +  "  and  "  +
object1.address.city);
System.out.println("Printing values from Object2:");
System.out.println("Name:"+object2.name + " RollNo:"+ object2.rollNo);
System.out.println("Address:"+  object2.address.pinCode  +  "  and  "  +
object2.address.city);
}
}
```

Output: *Printing values from Object1:Name:Ravi RollNo:1*
Address:505 and KUK
Printing values from Object2:Name:Ram RollNo:2
Address:505 and KUK



5.1.6 Klonimi i thellë (Deep Cloning)

Siç e patë në klonimin sipërfaqësor, lloji i referencës ndan referencën e objektit në vend që të krijojë një objekt të ri. Por në klonimin e thellë, bëhet kopjimi i gjithçkaje nga një objekt në një objekt tjetër (anëtarët primitivë dhe të të dhënave referuese). Në klonimin e thellë, objekti i klonuar është i pavarur nga objekti origjinal dhe bërja e ndryshimeve në objektin e klonuar nuk duhet të ndikojë në objektin origjinal. Nëse duam të krijojmë klonim të thellë të objektit tek objektit1 atëherë krijohet një kopje e re e secilit anëtar të referuar dhe këto referenca vendosen në objektin e objektit2. Anëtari referues i të dy objekteve nuk do të tregojë për të njëjtin objekt. Kjo do të thotë që çdo ndryshim i bërë në anëtarin e objektit të referuar në Objektin objekt 1 ose objekt2 do të pasqyrohet tek tjetri. *Si të arrihet klonimi i thellë?* Për të mbështetur klonimin e thellë, ne duhet të implementojmë një ndërfaqe Cloneable dhe duke tejkaluar metodën clone () në çdo anëtar të llojit të referencës që lidhet me hierarkinë tonë të objektit. Pastaj, ne i quajmë super.clone () këto metoda clone () në objektin tone të metodës clone () . Le të përpiqemi ta kuptojmë atë me një shembull

```
class Address implements Cloneable
{
int pinCode;
String city;
Address()
{ }
Address(int pinCode, String city)
```

```
{
this.pinCode = pinCode;
this.city = city;
}
public Object clone() throws CloneNotSupportedException
{
return super.clone();
}
}
class Student implements Cloneable
{
int rollNo;
String name;
Address address = new Address();
public Object clone() throws CloneNotSupportedException
{
return super.clone();
}
}
public class Data
{
public static void main(String[] args) throws CloneNotSupportedException
{
Student object1 = new Student();
object1.name = "Ravi";
object1.rollNo = 1;
object1.address = new Address(101, "CHD");
Student object2 = (Student) object1.clone();
object2.address = (Address) object1.address.clone();
System.out.println("Printing values from Object1:");
System.out.println("Name:"+object1.name + " RollNo:"+ object1.rollNo);
System.out.println("Address:"+  object1.address.pinCode  +  "  and  "  +
object1.address.city);
System.out.println("Printing values from Object2:");
System.out.println("Name:"+object2.name + " RollNo:"+ object2.rollNo);
System.out.println("Address:"+  object2.address.pinCode  +  "  and  "  +
object2.address.city);
System.out.println("Is both object are same in heap memory:" + (object1
== object2));
System.out.println("Is  both  Address  object  are  same  in  heap
memory:"+(object1.address == object2.address));
}
}
```

Output: *Printing values from Object1: Name:Ravi RollNo:1*
Address:101 and CHD
Printing values from Object2: Name:Ravi RollNo:1
Address:101 and CHD
Is both object are same in heap memory:false
Is both Address object are same in heap memory:false

Tani (object1.address == object2.address) do të vlerësojë false sepse, në metodën clone () të klasës Data, ne po klonojmë objektin e adresës dhe po ia caktojmë objektit të ri të klonuar.
Ndërsa shihni rezultatet e programit të mësipërm, të dy referencat tregojnë për objekte të ndryshme në një grumbull, madje të dy objektet kanë të njëjtat vlera të të dhënave.

5.1.7 Përparësitë e klonimit

1. Nëse duam të krijojmë një kopje të saktë të objektit pa klonim duhet të shkruajmë përsëri kodin. Me përdorimin e Klonimit, nuk kemi nevojë të shkruajmë kode të përsëritura. Thjesht duhet të implementojmë ndërfaqen dhe të anashkalojmë metodën e klonit () në klasë.
2. Le të themi që projekti juaj është zhvilluar tashmë dhe doni të krijoni një kopje të objektit për disa ndryshime. Atëherë klonimi do të jetë mënyra më e lehtë dhe më efikase e kopjimit të objekteve.
3. Klonimi është mënyra më e shpejtë për të kopjuar një grup.
4. Më efektive sesa konstruktorët e Kopjimit. Në konstruktorin e kopjimit, ne duhet të kopjojmë të gjitha të dhënat në mënyrë të qartë. Do të thotë që ne duhet të caktojmë të gjithë anëtarët e të dhënave (Fushat) e klasës në konstruktor. Por duke përdorur metodën clone (), ne mund ta bëjmë këtë punë brenda 4 ose 5 rreshtave.

5.2. Stringjet

5.2.1 Klasa String

Një string është një sekuencë karakteresh. Në shumë gjuhë, stringjet janë trajtuar si një tabelë (array) karakteresh, por në Java nj ë string është një objekt. Klasa String ka 11 konstruktorë dhe më shumë se 40 metoda për manipulimin e stringjeve. Jo vetëm që është shumë e përdorshme në programim, por kjo klasë është një mënyrë e mirë për të mësuar klasat dhe objektet.

5.2.2 Krijimi i një objekti String

Ju mund t ë krijoni nj ë objekt string nga një literal string-u ose nga një tabelë karakteresh. Për të krijuar një string nga një literal string-u përdoret sintaksa:

```
String emriVarRef = new String(literalString-u);
```

literalString-u është nj ë sekuencë karakteresh të futura midis thonjëzave dyshe. Statement-i i mëposhtëm krijon një objekt String mesazh për literalin string "Po mësojmë stringjet".

```
String mesazh = new String("Po mësojmë stringjet");
```

Java i trajton literalet e stringjeve si objekte String. Kështu, statement-i i mëposhtëm është i vlefshëm:

```
String mesazh = "Po mësojmë stringjet";
```

Ju mund të krijoni një string nga një tabelë karakteresh. Për shembull statement-et e mëposhtme krijojnë stringun “E premte”:

```
char[] tabKarakteresh = {'E', ' ', 'p', 'r', 'e', 'm', 't',
'e'}; String mesazhi = new String(tabKarakteresh);
```

Një variabël String mban një referencë të një objekti String që ruan një vlerë string. Fjala string shpesh përdoret për tiu referuar një variabli String, një objekti String ose një vlere string.

5.2.2 Stringjet e pandryshueshme

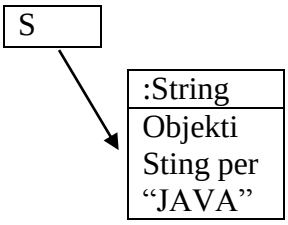
Një objekt String është i pandryshueshëm sepse përmbajtjet e tij nuk mund të ndryshohen. A i ndryshon kodi i mëposhtëm përmbajtjet e stringut?

```
String s = "Java";
s = "HTML";
```

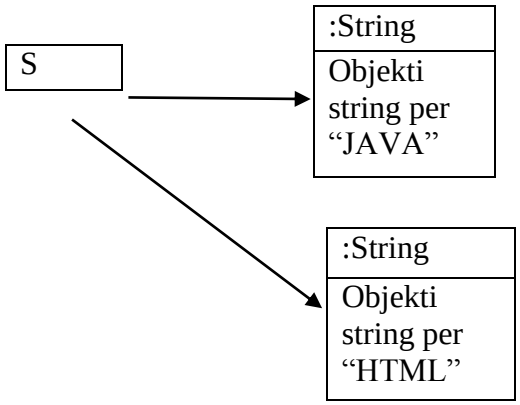
Përgjigjja është jo. Statement-i i parë krijon një objekt String me përmbajtje “Java” dhe e cakton referencën e tij tek s. Statement-i i dytë krijon një objekt të ri String me përmbajtje “HTML” dhe e cakton referencën e tij tek s. Objekti i parë String vazhdon të ekzistojë pas caktimit, por nuk mund

të aksesohet më, sepse variabli s tashmë shënjon tek objekti i dytë, siç tregohet në figurën e mëposhtme:

Pasi ekzekutohet String s="java"



Pasi ekzekutohet s="HTML"



5.2.3 Krahasimi i stringjeve

Klasa String ofron metoda për të krahasuar stringje. Ato metoda janë paraqitur më poshtë:

java.lang.St
ring +equals(s1: String): boolean
+equalsIgnoreCase(s1: String): boolean
+compareTo(s1: String): int
+compareToIgnoreCase(s1: String): int
+regionMatches(index: int, s1: String, s1Index: int, len: int): boolean
+regionMatches(ignoreCase: boolean, index: int, s1: String, s1Index: int, len: int): Boolean
+startsWith(prefix: String): boolean
+endsWith(suffix: String): boolean

Kthen true nëse stringu që krahasohet është i barabartë me stringun s1.

Kthen true nëse stringu që krahasohet është i barabartë me stringun s1, kjo metodë është jo case sensitive.

Kthen një numër >0, =0, ose <0 nëse stringu që krahasohet është më i madh, ose i barabartë ose më i vogël se string-u s1.

Njëlloj si compareTo, por nuk merret parasysh gërma e madhe ose e vogël.

Kthen true nëse zona e specifikuar e stringut përputhet ekzaktësisht me zonën e specifikuar në stringun s1. Njëlloj si metoda më sipër ku specifikohet nëse merret parasysh dallimi i gërmës së madhe nga ajo e vogël.

Kthen true nëse stringu fillon me parashtesën e specifikuar.

Kthen true nëse stringu mbaron me prapashtesën e specifiku

Si krahasohet përmbajtja e dy stringjeve? Ju mund të përdorni operatorin ==, si më poshtë:

```
if (string1 == string2)
System.out.println("string1 dhe string2
jane i njejti objekt"); else
System.out.println("string1 dhe string2
jane objekte te ndryshme");
```

Gjithsesi, operatori == kontrollon nëse string1 dhe string2 referojnë ju nëse tek i njëjti objekt, ai nuk ju tregon stringjet kanë të njëjtat përmbajtje. Për këtë ju duhet të përdorni metodën equals. Kështu kodi i mëposhtëm mund të përdoret për të krahasuar dy stringje:

```
if (string1.equals(string2))
System.out.println("string1 dhe string2
kane permbajtje te njejta"); else
System.out.println("string1 dhe string2
nuk jane njesoj");
```

Për shembull, statementet e mëposhtme shfaqin true dhe më pas false:

```
String s1 = new String("Semestri I"); String s2 = "Semestri I";
String s3 = "Semestri II"; System.out.println(s1.equals(s2)); // true
System.out.println(s1.equals(s3)); // false
```

Metoda compareTo gjithashtu mund të përdoret për të krahasuar dy stringje. Për shembull, shikoni kodin e mëposhtëm:

```
s1.compareTo(s2);
```

Metoda kthen vlerën 0 nëse s1 është i njëjtë me s2, një vlerë më të vogël se 0 nëse s1 është më i vogël se s2 dhe një vlerë më të madhe se 0 nëse s1 është më i madh se s2. P ër shembull, supozojmë se s1 është"abc" dhe s2 është"abg". Fillimisht krahasohen dy karakteret e para (a me a) nga s1 dhe s2. Meqenëse janë të njëjta, krahasohen dy karakteret e dyta (b me b). Meqenëse janë të njëjta, krahasohen dy karakteret e treta (c me g). Meqen ëse karakteri c është 4 më i vogël se g (c është shkronja e 3 në alfabetin amerikan kurse g është e 7), krahasimi kthen -4.

Nëse ju do të përdorni operatorët e krahasimitsi >, >=, < ose <= për të krahasuar stringje do të ndodhin gabime sintaksore. Në vend të tyre duhet të përdorni s1.compareTo(s2).

Klasa String ofron gjithashtu metodat equalsIgnoreCase, compareToIgnoreCase dhe regionMatches për të krahasuar stringjet. Metodat equalsIgnoreCase dhe compareToIgnoreCase janë si metodat equals dhe compareTo përkatësisht, vetëm se injorojnë shkronjat e mëdha dhe të vogla kur krahasojnë stringjet. Metoda regionMatches krahason pjesë të dy stringjeve për barazi. Ju mund të përdorni str.startsWith(parashtesa) për të kontrolluar nëse stringu str fillon me nj ë parashtesë specifike dhe str.endsWith(prapashtesë) për të kontrolluar nëse stringu str mbaron me një prapashtesë specifike.

5.2.4 Gjatësia e stringjeve

Ju mund të gjeni gjatësinë e një stringu duke thirrur metodën length(). Për shembull, mesazh.length()kthen gjatësinë e stringut mesazh. length është një metodë në klasën String dhe një cilësi e nj ë objekti tabelë. Kështu ju mund të përdorni s.length() për të gjetur numrin e karaktereve në një string s dhe a.length për të marrë numrin e elementëve në një tabelë a.

5.2.5 Kthimi i një karakteri në varësi të indeksit

Metoda s.charAt(indeksi)mund të përdoret për të kthyer një karakter specifik në një string s, ku indeksi është midis 0 dhe s.length-1. Për shembull, mesazh.charAt(0)kthen karakterin P, siç tregohet në figurë:

Indekset:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

mesazh	P	o		m	ë	s	o	j	M	ë		s	t	R	i	n	g	j	e	t

5.2.6 Bashkimi i stringjeve

Ju mund të përdorni metodën concat për të bashkuar dy stringje. Për shembull, statement-i i mëposhtëm
bashkon stringjet s1 dhe s2 tek s3:

```
String s3 = s1.concat(s2);
```

Gjithashtu ju mund të përdorni shenjën + për të lidhur dy ose më shumë stringje. Kështu statement-i i mësipërm është ekuivalent me:

```
String s3 = s1 + s2;
```

Kodi i mëposhtëm kombinon stringjet mesazh, "në" dhe "leksionin 8" në një string:

```
String mesazhStr = mesazh + "në" + "leksionin 8";
```

Shenja + mund të përdoret dhe për të lidhur një numër me një string. Në këtë rast, numri konvertohet në një string dhe më pas bëhet bashkimi i stringjeve.

5.2.7 Marrja e nënstringjeve

Ju mund të merrni një karakter të vetëm nga një string duke përdorur metodën charAt. Ju mund të merrni një nënstring nga një string duke përdorur metodën substring në klasën String. Shembull:

```
mesazh.substring(11); //marrim nënstringun “stringjet”
```

Statement-i i mësipërm kthen një nënstring të stringut mesazh duke filluar nga karakteri me indeks 11 dhe duke vazhduar deri në fund të stringut.

```
mesazh.substring(0,2); //marrim nënstringun “Po”
```

Statement-i i mësipërm kthen një nënstring të stringut mesazh duke filluar nga karakteri me indeks 0 dhe duke vazhduar deri tek indeksi 2-1.

5.2.8 Konvertimi i stringjeve

Gjatë konvertimit të një stringu kthehet një string i ri i derivuar nga stringu origjinal duke mos ndryshuar stringun origjinal. Për të konvertuar stringjet përdoren metodat toLowerCase, toUpperCase dhe trim. Metoda toLowerCase kthen një string të ri duke konvertuar gjithë karakteret në string në shkronja të mëdha. Metoda toUpperCase kthen një string të ri duke konvertuar gjithë karakteret në string në shkronja të vogla. Metoda trim kthen një string të ri duke eliminuar karakteresh boshe(hapësirat) në fillim dhe në fund të string-ut.

Shembuj:

```
"Programe".toLowerCase(); //kthen nje string te ri, programe.
"Programe".toUpperCase(); //kthen nje string te ri, PROGRAME.
" Programe  ".trim(); //kthen nje string te ri, Programe.
```

5.2.9 Zëvendësimi i stringjeve

Klasa String ofron disa versione të metodës replace për të zëvendësuar një karakter ose një nënstring në një string me një karakter ose një nënstring të ri.
Metoda replace kthen një string që zëvendëson gjithë karakteret në string, që përputhen me karakterin e specifikuar, me një karakter të ri.

```
"Programe".replace('r','A');   //kthen      nje  string      te    ri,
                PAogAame.
```

```
System.out.println("Programe".replace('r', 'A')); //printon PAogAme
```

Metoda replaceFirst kthen nj ë string që zëvendëson nënstringun e parë në string, që përputhet me nënstringun e specifikuar, me një nënstring të ri.

```
"Programe".replaceFirst("r", "AB"); //kthen nje string te ri, PABograme.
```

Metoda replaceAll kthen një s tring që zëvendëson gjithë nënstringjet në string, që përputhen me nënstringun e specifikuar, me një nënstring të ri.

```
"Programe".replaceAll("r", "ABC"); //kthen nje string te ri,
PABCgABCame.
```

5.2.10 Ndarja e stringjeve

Metoda split mund të përdoret për të ndarë një string në pjesë bazuar në një ndarës specifik. Shembull:

```
String str = "Po-ndahet-stringu-ne-
pjese"; String[] pjeset = str.split("-
");
for (int i = 0; i < pjeset.length; i++){
System.out.print(pjeset[i] + " ");
}

Output:
Po ndahet stringu ne pjese
```

5.2.11 Shprehjet e rregullta

Ju mund të kontrolloni për përputhje, të zëvendësoni dhe të ndani një string duke përdorur një model (pattern). Kjo është një veçori njihet si *shprehjet e rregullta*. Le të nisim me metodën equals të klasës String. Metoda matches është shumë e ngjashme me metodën equals. Për shembull, dy statementet e mëposhtme vlerësohen si true.

```
"Java".matches("Java");
"Java".equals("Java");
```

Megjithatë metoda matches është më e fuqishme. Ajo mund të kontrollojë për përputhshmëri jo vetëm një string të fiksuar, por gjithashtu një grup stringjesh që ndjekin një model. Për shembull, statement-et e mëposhtme vlerësohen të gjitha si true.

```
"Javaeshte e thjeshte".matches("Java.*"); "Java eshte
    object oriented".matches("Java.*"); "Java eshte e
interpretueshme".matches("Java.*");
```

"Java.*" në statement-et e mësipërme është një *shprehje e rregullt*. Ai përshkruan një model string që fillon me Java dhe që ndiqet nga zero ose më shumë karaktere. Pra nënstring-u .* do të thotë zero ose më shumë karaktere.
Metodat replaceAll, replaceFirst dhe split mund të përdoren me nj ë shprehje të rregullt. P ër shembull, statement-i i mëposhtëm kthen një string të ri që zëvendëson $, + ose # në "a+b$#c" me stringun NNN.

```
String s = "a+b$#c".replaceAll("[$+#]","NNN");
System.out.println(s);
```

```
String[] tabeleStr =
"Java,C?C#,C++".split("[.,:;?]"); for (int i = 0;
i < tabeleStr.length; i++);
System.out.println(tokens[i]);
```

Shprehja e rregullt [.,:;?] specifikon një model që përputhet me .,,,:,; ose ?. Secili prej këtyre karaktereve është një kufi për ndarjen e stringut. Kështu output-i i kodit të mësipërm është:

Java
C
C#
C++

5.2.12 Gjetja e një karakteri ose nënstring-u në një string

Klasa String ofron disa metoda të mbingarkuara indexOf dhe lastIndexOf për gjetjen e një karakteri ose një nënstring-u në një string.

java.lang.String	
+indexOf(ch: char): int	Kthen indeksin e të parit karakter *ch* në string. Kthen -1 nëse ky karakter nuk ndodhet në string.
+indexOf(ch: char, ngaIndex: int): int	Kthen indeksin e të parit karakter *ch* në string pas *ngaIndex*. Kthen -1 nëse ky karakter nuk ndodhet në string.
+indexOf(s: String): int	Kthen indeksin e të parit string *s* në string. Kthen -1 nëse *s* nuk ndodhet në string.
+indexOf(s: String, ngaIndeks: int): int	Kthen indeksin e të parit string *s* në string pas *ngaIndeks*. Kthen - 1 nëse *s* nuk ndodhet në string.
+lastIndexOf(ch: int): int	Kthen indeksin e karakterit të fundit *ch* në string. Returns -1 nëse karakteri ch nuk gjëndet në string.
+lastIndexOf(ch: int, ngaIndeks: int): int	Kthen indeksin e karakterit të fundit *ch* në string para *ngaIndeks*. Kthen -1 nëse nuk gjendet.
+lastIndexOf(s: String): int	Kthen indeksin e stringut të fundit *s* në string. Kthen -1 nëse stringu *s* nuk gjendet në string.
+lastIndexOf(s: String, ngaIndeks: int): int	Kthen indeksin e stringut të fundit *s* në string para *ngaIndeks*. Kthen -1 nëse stringu *s* nuk gjendet.

Shembull:

```
System.out.println("Gezuar festat!".indexOf('G'));//kthen 0.
System.out.println("Gezuar festat!".indexOf('a')); //kthen 4.
System.out.println("Gezuar festat!".indexOf('a', 5));// kthen 11.
System.out.println("Gezuar festat!".indexOf("uar"));// kthen 3.
System.out.println("Gezuar festat!".indexOf("festat", 5));// kthen 7.
System.out.println("Gezuar festat!".indexOf("Festat", 5));// kthen -1.
```

```
System.out.println("Gezuar festat!".lastIndexOf('G'));// kthen 0.
System.out.println("Gezuar festat!".lastIndexOf('a'));// kthen 11.
System.out.println("Gezuar festat!".lastIndexOf('a', 5));// kthen 4.
System.out.println("Gezuar festat!".lastIndexOf("festat"));// kthen 7.

System.out.println("Gezuar festat!".lastIndexOf("festat", 5));// kthen -
1
```

5.2.13 Konvertimi midis stringjeve dhe tabelave

Stringjet nuk janë tabela, por një string mund të konvertohet në një tabelë dhe e kundërta. Për të konvertuar një string në një tabelë karakteresh përdoret metoda toCharArray. Për shembull statement-i i mëposhtëm konverton stringun "Java" në një tabelë:

```
char[] tabKarakt = "Java".toCharArray();
```

Kështu tabKarakt[0] është 'J', tabKarakt[1] është 'a', tabKarakt[2] është 'v', dhe tabKarakt[3] është 'a'. Ju mund të përdorni metodën getChars(int fillimBurim, int fundBurim, char[] dest, int fillimDest) për të kopjuar një nënstring të një stringu nga indeksi fillimBurim tek fundBurim-1 në një tabelë karakteresh dest duke filluar nga indeksi fillimDest.

Për shembull kodi i mëposhtëm kopjon një nënstring “3456” në stringun “AB3456” nga indeksi 2 tek indeksi 6-1 në një tabelë karakteresh dest duke filluar nga indeksi 4.

```
char[] dest = {'J', 'A', 'V', 'A', '1', '3',
'0', '1'}; "AB3456".getChars(2, 6, dst, 4);
Kështu dest bëhet {'J', 'A', 'V', 'A', '3',
'4', '5', '6'}.
```

Për të konvertuar një tabelë karakteresh në një string përdoret konstruktori String(char[]) ose metoda valueOf(char[]). Për shembull, statement-i i mëposhtëm ndërton një string nga një tabelë duke përdorur konstruktorin String.

```
String str = new String(new char[]{'J', 'a', 'v', 'a'});
```

Statement-i pasues ndërton një string nga një tabelë duke përdorur metodën valueOf.

```
String str = String.valueOf(new char[]{'J', 'a', 'v', 'a'});
```

5.2.14 Konvertimi i karaktereve dhe vlerave numerike në stringje

Metoda statike valueOf mund të përdoret për të konvertuar një tabelë karakteresh në një string. Ekzistojnë disa versione të mbingarkuara të metodës valueOf që mund të përdoren për të konvertuar një karakter dhe vlerat numerike në stringje me tipe të ndryshme parametrash si char, double, long, int dhe float., siç tregohet n
figurën e mëposhtme:

java.lang.String	
+valueOf (c : char) String	Kthen një string që përbëhet nga karakteri c.
+valueOf(data: char[]): String	Kthen një string që përbëhet nga karakteret në tabelë.
+valueOf(d: double): String	Kthen një string që paraqet vlerën double.
+valueOf(f: float): String	Kthen një string që paraqet vlerën float.
+valueOf(i: int): String	Kthen një string që paraqet vlerën int.
+valueOf(l: long): String	Kthen një string që paraqet vlerën long.
+valueOf(b: boolean): String	Kthen një string që paraqet vlerën boolean-e.

Për shembull, për të konvertuar një vlerë double 2.51 në një string, përdoret String.valueOf(2.51). Vlera e kthyer është një string që përbëhet nga karakteret '2', '.', '5', dhe '1'.
Përdorni Double.parseDouble(str) ose Integer.parseInt(str) për të konvertuar një string str në një vlerë double ose në një vlerë int.

5.2.15 Klasa Character

Klasa Character ndodhet në paketën java.lang. Kjo klasë bën të mundur që vlerat e tipit char të trajtohen si objekte. Klasa Character ka nj ë konstruktor dhe disa metoda për të vendosur kategorinë e një karakteri (shkronj ë e madhe, shkronjë e vogël, shifër, etj.) dhe për konvertimin e karaktereve nga shkronjë e madhe në shkronjë të vogël dhe e kundërta. Ju mund të krijoni një objekt Character nga një vlerë char. Për shembull, statement-i i mëposhtëm krijon një objekt Character nga një karakter 'a'.

```
Character karakter = new Character('a');
```

Metoda charValue kthen vlerën e karakterit të mbështjellë në objektin Character. Metoda compareTo krahason këtë karakter me një karakter tjetër dhe kthen një numër të plotë që është diferenca midis Unicode-eve të këtij karakteri dhe karakterit tjetër. Metoda equals kthen true vetëm nëse dy karakteret janë të njëjta. Për shembull, supozoni se objKarakter është new Character ('b'):

```
System.out.println(objKarakter.compa        Character('a')));//k
reTo(new                                    then                    1
System.out.println(objKarakter.compa        Character('b')));//k
reTo(new                                    then                    0
System.out.println(objKarakter.compa       Character('c')));//k
reTo(new                                    then                   -1

System.out.println(objKarakter.compareTo(new Character('d')));//kthen -2

System.out.println(objKarakter.equals(new Character('b')));//kthen true
System.out.println(objKarakter.equals(new Character('d')));// kthen false
```

Në figurën e mëposhtme janë paraqitur metodat e klasës Character që mund tëpërdoren për manipulimin e një karakteri.

java.lang.Character	
+Character(vlera: char)	Ndërtohet objekt karakter me vlerë të tipit char.
+charValue(): char	Kthen vlerën karakter nga ky objekt.
+compareTo(karakterTjeter: Character): int	Krahason këtë karakter me një tjetër.
+equals(karakterTjeter: Character): Boolean	Kthen true nëse karakteri është i barabartë me një tjetër.
+isDigit(ch: char): boolean	Kthen true nëse karakteri i specifikuar është një shifër.
+isLetter(ch: char): boolean	Kthen true nëse karakteri i specifikuar është një gërmë.
+isLetterOrDigit(ch: char): Boolean	Kthen true nëse karakteri i specifikuar është një shifër ose gërmë.
+isLowerCase(ch: char): boolean	Kthen true nëse karakteri është gërmë e vogël.
+isUpperCase(ch: char): boolean	Kthen true nëse karakteri është gërmë e madhe.

[redacted]	
+toLowerCase(ch: char): char	Kthen karakterin si gërme të vogël.
[redacted]	
+toUpperCase(ch: char): char	Kthen karakterin si gërme të madhe.

```
System.out.println(objKarakter.charValue());//
kthen b
System.out.println(Character.isDigit(objKarakt
er));                                          //kthen          false
System.out.println(Character.isLetter(objKarak
ter));                                          //kthen          true
System.out.println(Character.isLetterOrDigit(o
bjKarakter));//kthen                                           True
System.out.println(Character.isLowerCase(objKa
rakter));                                        //kthen         True
System.out.println(Character.isUpperCase(objKa
rakter));                                        //kthen         False
System.out.println(Character.toLowerCase(objKa
rakter));//kthen                                                B
System.out.println(Character.toUpperCase(objKa
rakter));//kthen B
```

Ushtrimi 1

Shkruani një program që kontrollon nëse një fjal ë e futur si input nga përdoruesi është palindromë. Nj ë palindromë është një fjalë që lexohet njësoj nga fillimi në fund dhe nga fundi në fillim, si fjala “radar”.

```
import java.util.Scanner;
public class Palindrome {

 public static void main(String[] args) {
           //krijon nje Scanner
 Scanner input = new Scanner(System.in);

 System.out.print("Fusni nje string: ");
 String s = input.nextLine();

 if(eshtePalindrome(s))

    System.out.println(s + " eshte nje palindrome"); else
    System.out.println(s + " nuk eshte nje palindrome");
 }

 public static boolean eshtePalindrome(String
 s ) { int karakteriPare = 0; int
 karakteriFundit = s.length()-1;

 while(karakteriPare<karakteriFundit){
  if(s.charAt(karakteriPare)!=
  s.charAt(karakteriFundit)) return false;//nuk eshte
  palindome

  karakteriPare++;
  karakteriFundit--;
 }
```

```
    return true;
  }
}
```

Ushtrimi 2
Shkruani një program që gjeneron një password për një student duke përdorur inicialet e tij dhe moshën.

```
public class krijoPassword {

      public static void main(String[] args) {
            String emri = "Blerta";
             String mbiemri = "Blushi";

             int mosha = 22;

             String inicialet = emri.substring(0,1) +
             mbiemri.substring(0,1);

             //shto    moshen    pasi ti    kesh     kthyer inicialet   ne
                 shkronja te    vogla      String   password    =

             inicialet.toLowerCase() + mosha;

             System.out.println("Passord-i juaj = " + password);
      }
}

Ky program afishon:
Passord-i juaj = bb22
```

5.3 The JDK LinkedList class

Lista e Lidhur është një pjesë e koleksionit e pranishme në paketën java.util. Kjo klasë është një implementim i strukturës së të dhënave LinkedList e cila është një strukturë lineare e të dhënave ku elementet nuk ruhen në vende të afërta dhe çdo element është një objekt i veçantë me një pjesë të të dhënave dhe një pjesë të adresës. Elementet lidhen duke përdorur pointera dhe adresa. Secili element njihet si nyje. Për shkak të dinamikës dhe lehtësisë së futjeve dhe fshirjeve, ato preferohen mbi vargjet. Ai gjithashtu ka disa disavantazhe si nyjet nuk mund të arrihen direkt në vend që të duhet të fillojmë nga koka dhe të ndjekim përmes lidhjes për të arritur në një nyje që dëshirojmë të kemi.

Le ta shohim me nje shembull:

```
import java.util.*;
public class Test {
    public static void main(String args[])
    {
        // Creating object of the
        // class linked list
        LinkedList<String> ll
            = new LinkedList<String>();
        // Adding elements to the linked list
        ll.add("A");
        ll.add("B");
        ll.addLast("C");
```

```
        ll.addFirst("D");
        ll.add(2, "E");
        System.out.println(ll);

        ll.remove("B");
        ll.remove(3);
        ll.removeFirst();
        ll.removeLast();
        System.out.println(ll);
    }
}
```

Output:

```
[D, A, E, B, C]

[A]
```

Kryerja e operacioneve të ndryshme në ArrayList

Le të shohim se si të kryejmë disa operacione themelore në LinkedList.

1. Shtimi i elementeve: Në mënyrë që të shtojmë një element në një ArrayList, mund të përdorim metodën add (). Kjo metodë është e mbingarkuar për të kryer operacione të shumëfishta bazuar në parametra të ndryshëm. Ata janë:

a. add (Object): Kjo metodë përdoret për të shtuar një element në fund të LinkedList.

b. add (int index, Object): Kjo metodë përdoret për të shtuar një element në një indeks specifik në LinkedList.

```
// Java program to add elements
// to a LinkedList
import java.util.*;
public class GFG {
    public static void main(String args[])
    {
        LinkedList<String> ll = new LinkedList<>();
        ll.add("Geeks");
        ll.add("Geeks");
        ll.add(1, "For");
        System.out.println(ll);
    }
}
```

Output:

```
[Geeks, For, Geeks]
```

2. Ndryshimi i elementeve: Pas shtimit të elementeve, nëse dëshirojmë të ndryshojmë elementin, mund të bëhet duke përdorur metodën set (). Meqenëse një LinkedList indeksohet, elementi të cilin dëshirojmë ta ndryshojmë referohet nga indeksi i elementit. Prandaj, kjo metodë merr një indeks dhe elementin e azhurnuar i cili duhet të futet në atë indeks.

```
// Java program to change elements
// in a LinkedList
import java.util.*;
public class GFG {
    public static void main(String args[])    {
        LinkedList<String> ll = new LinkedList<>();
        ll.add("Geeks");
        ll.add("Geeks");
        ll.add(1, "Geeks");
        System.out.println("Initial LinkedList " + ll);
        ll.set(1, "For");
```

```
        System.out.println("Updated LinkedList " + ll);    }   }
```

Output:

```
Initial LinkedList [Geeks, Geeks, Geeks]

Updated LinkedList [Geeks, For, Geeks]
```

3. Heqja e elementeve: Për të hequr një element nga një LinkedList, ne mund të përdorim metodën remove (). Kjo metodë është ngarkuar për të kryer operacione të shumëfishta bazuar në parametra të ndryshëm. Ata janë: remove (Object): Kjo metodë përdoret për të hequr thjesht një objekt nga LinkedList. Nëse ka shumë objekte të tilla, atëherë dukuria e parë e objektit hiqet. remove (int index): Meqenëse indeksohet një LinkedList, kjo metodë merr një vlerë të plotë e cila thjesht heq elementin e pranishëm në atë indeks specifik në LinkedList. Pas heqjes së elementit, të gjithë elementët zhvendosen majtas për të mbushur hapësirën dhe azhurnohen indekset e objekteve.

```
// Java program to remove elements
// in a LinkedList
import java.util.*;
public class GFG {
    public static void main(String args[])
    {
        LinkedList<String> ll = new LinkedList<>();
        ll.add("Geeks");
        ll.add("Geeks");
        ll.add(1, "For");
        System.out.println(
            "Initial LinkedList " + ll);
        ll.remove(1);
        System.out.println(
            "After the Index Removal " + ll);
        ll.remove("Geeks");
        System.out.println(
            "After the Object Removal " + ll);
    }
}
```

Output:

```
Initial LinkedList [Geeks, For, Geeks]

After the Index Removal [Geeks, Geeks]

After the Object Removal [Geeks]
```

Crystal
Talent Solutions