ជំពូកទី ៩ : អំពី Strings

១. ទំរង់ constructors នៃ String

String class មានទំរង់ constructors មួយចំនួន គឺ :

ឧទាហរណ៍ : ដើម្បីបង្កើត String ទទេមួយ គេសរសេរ :

```
String s = new String();
```

ដើម្បីបង្កើត strings ដែលមានការកំណត់តំលៃដំបូង

គេអាចប្រើ constructor :

String(char *chars*[])

ឧទាហរណ៍ : char chars[] = {'a', 'b', 'c'};

String s = new String(chars); // s = "abc“

លើសពីនេះ គេអាចបង្កើត strings តាមការកំណត់ចំនួនតួអក្សរ មានក្នុង array មួយដោយប្រើទំរង់ constructor ដូចខាងក្រោមនេះ :

String(char *chars*[], int *startIndex*, int *numChars*)

ក្នុងនេះ *startIndex* ជា index ដែលបានកំណត់យកជាចំនុចចាប់ផ្តើម ហើយ *numChars* ជាចំនួនតួអក្សរដែលត្រូវការ ។

ឧទាហរណ៍ : char chars[] = {'a', 'b', 'c', 'd', 'e', 'f'};

String s = new String(chars, 2, 3);

// s = "cde"

គេក៏អាចបង្កើត object នៃ String ពី object នៃ String មួយទៀតបានដែរ ដោយអនុវត្តតាមទំរង់ constructor ដូចខាងក្រោម :

String(String *strObj*)

ដែលក្នុងនេះ *strObj* ជា object នៃ String ។

ឧទាហរណ៍ :

```
// Construct one String from another.
class MakeString {
  public static void main(String a[]){
    char c[] = {'J', 'a', 'v', 'a'};
    String s1 = new String(c);
    String s2 = new String(s1);
    System.out.println(s1);
    System.out.println(s2);
  }
}
```

ទោះបីជា Java មានការប្រើប្រភេទទិន្នន័យ char ដែលមានទំហំ 16 bits ក៏ដោយ វាបានផ្តល់នូវទំរង់ constructors ដែលមានការប្រើជាមួយ byte array បានដែរ។ ទំរង់ constructors មានដូចខាងក្រោម :

String(byte *asciiChars*[])
String(byte *asciiChars*[], int *startIndex*, int *numChars*)

កម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រើទំរង់ constructors ទាំងនេះ:

```
// Construct string from subset of char array.
class SubStringCons {
  public static void main(String args[]) {
    byte ascii[] = {65, 66, 67, 68, 69, 70 };
    String s1 = new String(ascii);
    System.out.println(s1);
    String s2 = new String(ascii, 2, 3);
    System.out.println(s2);
  }
}
```

២. ប្រវែង String

ប្រវែង string មួយគឺជាចំនួនតួអក្សរដែលវាមាន ។ ដើម្បីទទួលបានតំលៃនេះ យើងប្រើ length() method ដែលមានទំរង់ដូចខាងក្រោម :

int length()

ឧទាហរណ៍ : code ខាងក្រោមនេះបោះតំលៃ 3 ព្រោះវាមានចំនួន 3 តួអក្សរនៅក្នុង string ។

```
char chars[] = {'a', 'b', 'c'};
String s = new String(chars);
System.out.println(s.length());
```

៣. ការប្រើ String Literals

គេអាចប្រើ string literal ដើម្បីកំណត់តំលៃដំបូងអោយ object នៃ String បាន។

ឧទាហរណ៏ :

```
String s = "abc";  // use string literal
```

ហេតុនេះ គេក៏អាចប្រើវាជាមួយ length() method ហើយទទួលបានតំលៃ 3 ដូចឧទាហរណ៏ខាងលើដែរ ។

```
System.out.println("abc".length());
```

៤. ការភ្ជាប់ String

ជាទូទៅភាសា Java មិនអនុញ្ញាតអោយប្រើសញ្ញានព្ជន្ធលេខជាមួយ object នៃ String ឡើយ។ លើកលែងតែសញ្ញា + ប៉ុណ្ណោះដែលប្រើសំរាប់ភ្ជាប់ strings ពីរអោយលទ្ធផលជា object នៃ String ។

ឧទាហរណ៍ : code ខាងក្រោមបង្ហាញការភ្ជាប់ stringចំនួនបី ។

```
String age = "9";
String s = "He is " + age + " years old.";
System.out.println(s);
```

អនុវត្តដាក់ស្តែងអំពីការភ្ជាប់ strings ជាទូទៅគេប្រទះ
ឃើញមាននៅក្នុង code ដែលមានការប្រើ strings រាងៗ
ដោយគេត្រូវបំបែកជាចំណែកតូចៗដាក់ក្នុងសញ្ញា " "
និងប្រើសញ្ញា + ដើម្បីភ្ជាប់វា។

```
// Using concatenation to prevent long lines.
class ConCat {
  public static void main(String args[]) {
    String longStr = "This could have been " +
          "a very long line that would have " +
 "wrapped around. But string concatenation "
 +   "prevents this.";
    System.out.println(longStr);
  }
}
```

៥. ការភ្ជាប់ string ជាមួយនឹងប្រភេទទិន្នន័យផ្សេងទៀត

គេអាចភ្ជាប់ strings ជាមួយនឹងប្រភេទទិន្នន័យបានទៀត ដែរ ។ ឧទាហរណ៍ខាងក្រោមនេះបង្ហាញពីការប្រើលក្ខណៈនេះ ។

```
int age = 9;
String s = "He is " + age + " years old.";
System.out.println(s);
```

នៅក្នុងករណីនេះ age ជា int ប៉ុន្តែវាអោយលទ្ធផលដូចលើកមុនដែរ ។ នេះមកពីតំលៃ int នៅក្នុង age ត្រូវបំលែងដោយស្វ័យប្រវត្តទៅជា string ដែលតាងអោយ object នៃ String ។ បន្ទាប់មក string នេះធ្វើការភ្ជាប់ដូចលើកមុន ។

ទោះជាយ៉ាងណាក៏ដោយ យើងគួរតែយកចិត្តទុកទៅលើការប្រើប្រភេទទិន្នន័យផ្សេងៗភ្ជាប់ជាមួយ នឹង string ។ ព្រោះវាអាចអោយលទ្ធផលស្មានមិនដល់ ។ ចូរយើងពិនិត្យលើ code ខាងក្រោមនេះ :

```
String s = "four: " + 2 + 2;
System.out.println(s);
```

វានឹងអោយលទ្ធផលជា : `four: 22`

មិនមែនដូចយើងស្មានថា : `four: 4`

នោះទ្បើយ ។ នេះមកពីអត្ថិភាពនៃសញ្ញានព្ទន្ទលេខដែលត្រូវគិត

"four" ជាមួយនឹងលេខ ២ មុន។ ដូចនេះ ដើម្បីបានលទ្ធផល four:4 នោះ គេត្រូវសរសេរ :

```
String s = "four: "+(2+2);
```

៦. ការបំលែង String និងការប្រើ toString( )

កាលណា Java បំលែងប្រភេទទិន្នន័យទៅជា string ក្នុងពេលធ្វើការភ្ជាប់នោះ វានឹងមានការប្រើ method បំលែង string មួយក្នុងចំណោម overloaded methods ទាំងអស់ ដែលកំណត់ដោយ String។ Method នោះ គឺ valueOf()

ដែលវាមានលក្ខណៈ overloaded ចំពោះប្រភេទទិន្នន័យទំរង់ងាយ និងប្រភេទ Object ។ ចំពោះប្រភេទទិន្នន័យទំរង់ងាយ valueOf() នឹងអោយលទ្ធផលជា string នូវតំលៃរបស់វាដែលបានប្រើ ។ ចំពោះ object, valueOf() ហៅ toString() របស់ Object មកប្រើ ។ វាមានទំរង់ដូចខាងក្រោម:

String toString()

ដើម្បីយល់ពីលក្ខណៈនេះ ចូរសង្កេតនៅ code ខាងក្រោមនេះ:

៧. ការទាញយកតួអក្សរ *(Character Extraction)*

៧.១ ការប្រើ charAt()

ដើម្បីទាញយកតួអក្សរតែមួយគត់ពី String គេអាចប្រើ charAt() method ។ វាមានទំរង់ទូទៅ :

char charAt(int *where*)

ក្នុងនេះ *where* ជា index នៅត្រង់តួអក្សរដែលយើងចង់បាន ។ តំលៃ *where* ត្រូវតែវិជ្ជមាន ហើយកំនត់ទីតាំងណាមួយនៅក្នុង string ។

ឧទាហរណ៍ :

```
char ch;
ch = "abc".charAt(1);
```

កំណត់តំលៃ "b" ទៅអោយ ch ។

៧.២ ការប្រើ getChars()

បើសិនជាយើងត្រូវការដំណកយកតួអក្សរច្រើនក្នុងពេលតែមួយ គេប្រើ getChars() ។ វាមានទំរង់ទូទៅដូចខាងក្រោមនេះ :

```
void getChars(int sourceStart, int sourceEnd,
              char target[], int targetStart)
```

ដែលក្នុងនេះ *sourceStart* កំណត់អោយ index ចាប់ផ្តើម របស់ substring ហើយ *sourceEnd* កំណត់អោយ index ខាងចុងនៃ substring ។ ប៉ុន្តែតួអក្សរដែលទទួលបាននោះគឺ ពី *sourceStart* ទៅដល់ *sourceEnd–1* ប៉ុណ្ណោះ ។ array ដែលនឹងទទួលទុកដាក់នូវតួអក្សរនោះត្រូវបានកំណត់ដោយ *target* ។ index នៅក្នុង target ដែល substring នឹងត្រូវចំលងដាក់នោះគឺត្រូវឆ្លងតាម *targetStart* ។

កម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រើ getChars() ៖

```
class getCharsDemo {
  public static void main(String args[]) {
    String s = "This is a demo of the getChars method.";
    int start = 10;
    int end = 14;
    char buf[] = new char[end - start];
    s.getChars(start, end, buf, 0);
    System.out.println(buf);
  }
}
```

៧.៣ ការប្រើ getBytes()

គេមានវិធីមួយទៀតដែលអាចអោយគេផ្ទុកតួអក្សរនៅក្នុង array ជាប្រភេទ byte ។ method នោះ គឺ getBytes() ដែលវាប្រើលក្ខណៈបំលែងធម្មតាពីតួអក្សរទៅជា byte ។ ទំរង់របស់វាគឺ:

byte[ ] getBytes()

ឧទាហរណ៏ : កម្មវិធីខាងក្រោមនេះបង្ហាញពីការ getBytes().

៧.៤ ការប្រើ toCharArray()

បើយើងចង់បំលែងតួអក្សរទាំងអស់នៅក្នុង object នៃ String ទៅជាតួអក្សរនៅក្នុង array នោះ គេអាចប្រើ toCharArray() ។ វាអោយតំលៃជា array ផ្ទុកតួអក្សរទាំងអស់ដែលមាននៅក្នុង string ។ វាមានទំរង់ទូទៅ :

char[] toCharArray()

ឧទាហរណ៏ : កម្មវិធីខាងក្រោមបង្ហាញពីការប្រើ toCharArray() ។

```
class toCharsDemo {
    public static void main(String args[]) {
        String str = new String("I miss you!");
        char ch[];
        ch = str.toCharArray();
        for (int i=0; i < str.length(); i++)
            System.out.print(ch[i]);
    }
}
```

៨. ការប្រៀបធៀប String

៨.១ ការប្រើ equals() និង equalsIgnoreCase()

ដើម្បីធ្វើការប្រៀបធៀប String ពីរស្មើគ្នា គេប្រើ equals() ។ វាមានទំរង់ទូទៅដូចខាងក្រោម :

boolean equals(Object *str*)

ក្នុងនេះ *str* ជា object នៃ String ដែលត្រូវធ្វើការប្រៀបធៀបជាមួយនឹង object នៃ String មួយទៀត ។ វាអោយតំលៃ true កាលណា strings ទាំងពីរមានតួអក្សរដូចគ្នាតាមលំដាប់ដូចគ្នា ហើយផ្ទុយទៅវិញវាអោយតំលៃ false ។ ការប្រៀបធៀបនេះ

មានការប្រកាន់តួអក្សរតូចធំ ។

ដើម្បីធ្វើការប្រៀបធៀប ដោយមិនប្រកាន់តួអក្សរតូចធំ នោះ គេប្រើ equalsIgnoreCase() ។ វាមានទំរង់ទូទៅដូចខាងក្រោមនេះ :

boolean equalsIgnoreCase(String *str*)

ក្នុងនេះ *str* ជា object នៃ String ដែលត្រូវធ្វើការប្រៀបធៀបជាមួយនឹង object នៃ String មួយទៀត ។ វាអោយ តំលៃ true កាលណា strings ទាំងពីរមានតួអក្សរដូចគ្នាតាមលំដាប់ដូចគ្នា ហើយផ្ទុយទៅវិញវា អោយតំលៃ false ។

កម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រើ equals() និង equalsIgnoreCase() ។

```
class equalsDemo {
    public static void main(String args[]) {
        String s1 = "Hello";
        String s2 = "Hello";
        String s3 = "Good-bye";
        String s4 = "HELLO";
        System.out.println(s1 + " equals " + s2 + " -> " +
                                    s1.equals(s2));
        System.out.println(s1 + " equals " + s3 + " -> " +
                                    s1.equals(s3));
        System.out.println(s1 + " equals " + s4 + " -> " +
                                    s1.equals(s4));
        System.out.println(s1 + " equalsIgnoreCase "+s4+ " -> “
                                    + s1.equalsIgnoreCase(s4));
    }
}
```

៨.២ ការប្រើ regionMatches()

ដើម្បីធ្វើការប្រៀបធៀបនៅត្រង់ទីតាំងកំនត់របស់ String មួយទៅនឹងទីតាំងកំនត់នៅក្នុង String មួយទៀត គេត្រូវប្រើ regionMatches() ។ វាមានទំរង់ទូទៅដូចខាងក្រោមនេះ :

boolean regionMatches(int *startIndex*,
String *str2*, int *str2StartIndex*,
int *numChars*)

boolean regionMatches(boolean *ignoreCase*,
int *startIndex*, String *str2*,
int *str2StartIndex*, int *numChars*)

ក្នុងនេះ *startIndex* កំណត់អោយ index ដែលចាប់ផ្តើមត្រង់ទីតាំងនៃ String ត្រូវធ្វើការប្រៀបធៀប។ *str2* គឺជា String ទីពីរដែលធ្វើការប្រៀបធៀបជាមួយនឹងវា។ ហើយ index ដែលត្រូវយកសំរាប់ចាប់ផ្តើមប្រៀបធៀបរបស់ String ទីពីរគឺ *str2StartIndex*។ រីឯ *numChars* ជាប្រវែងនៃ substring ដែលត្រូវធ្វើការប្រៀបធៀប។ ហើយបើ *ignoreCase* កំណត់យក true នោះបានន័យថាវាមិនប្រកាន់ តួអក្សរតូចធំឡើយ ផ្ទុយទៅវិញវាប្រកាន់តួអក្សរតូចធំណាស់។

៨.៣ ការប្រើ startsWith() និង endsWith()

startsWith() method កំណត់នូវ string ដែលគេអោយទៅនឹងខាងដើម string ដែលគេចង់រក ។ endsWith() method កំណត់នូវ string ដែលគេអោយទៅនឹងខាងចុង string ដែលគេចង់រក ។ វាមានទំរង់ទូទៅដូចខាងក្រោម:

boolean startsWith(String *str*)
boolean endsWith(String *str*)

ក្នុងនេះ *str* គឺជា string ដែលត្រូវផ្ទៀងផ្ទាត់រក ។ បើវាត្រូវគ្នានោះវានឹងអោយតំលៃ true ផ្ទុយទៅវិញ វាអោយតំលៃ false ។

ឧទាហរណ៍ : "Foobar".endsWith("bar");

និង "Foobar".startsWith("Foo");

ទាំងពីរនេះសុទ្ធតែអោយតំលៃ true ។

ទំរង់មួយទៀតរបស់ startsWith() គឺ :

boolean startsWith(String *str*, int *startIndex*)

ក្នុងនេះ *startIndex* កំណត់អោយ index ដែលត្រូវចាប់ផ្តើមរកនៅក្នុង string នោះ ។

ឧទាហរណ៍ : "Foobar".startsWith("bar", 3);

វានឹងអោយតំលៃ true ។

៨.៤ ភាពខុសគ្នារវាង equals() និង ==

វាមានសារៈសំខាន់ណាស់ដើម្បីដឹងបានថា equals() method និងសញ្ញាណនព្ធន្ត == ធ្វើការ ប្រតិបត្តិផ្សេងគ្នា ។ equals() method ធ្វើការប្រៀបធៀបតួអក្សរនៅក្នុង object នៃ String ។

រីឯ == ធ្វើការប្រៀបធៀប objects ពីរ ដើម្បីមើលឃើញថា តើវាជា objects ដូចគ្នាដែរឬទេ។ កម្មវិធីខាងក្រោមនេះបង្ហាញនូវរបៀប objects ពីររបស់ String ផ្សេងគ្នាហើយបានផ្ទុកតួអក្សរដូចគ្នា តែការប្រៀបធៀប objects ទាំងពីរនេះមិនដូចគ្នាឡើយ។

កម្មវិធីខាងក្រោមនេះបង្ហាញនូវរបៀប objects ពីររបស់ String ផ្សេងគ្នាហើយបានផ្ទុកតួអក្សរដូចគ្នា តែការប្រៀបធៀប objects ទាំងពីរនេះមិនដូចគ្នាឡើយ។

```
// equals() vs ==
class EqualsNotEqualTo {
  public static void main(String args[]) {
    String s1 = "Hello";
    String s2 = new String(s1);
    System.out.println(s1 + " equals " + s2 +
                          " -> " +  s1.equals(s2));
    System.out.println(s1 + " == " +s2+
                          " -> "+(s1 == s2));
  }
}
```

៨.៥ ការប្រៀប compareTo()

ជាធម្មតា ការដឹងបានអំពី strings ពីរដូចគ្នានោះវាមិនគ្រប់គ្រាន់ឡើយ ។ ចំពោះការរៀបតាមលំដាប់ យើងត្រូវការដឹងថាតើមួយណាតូចជាង, មួយណាស្មើគ្នា ឬធំជាងបន្តបន្ទាប់គ្នា ។ បើ string តូចជាងនោះវាត្រូវរៀបខាងដើមតាមលំដាប់វចនានុក្រម ហើយធំជាងត្រូវរៀបក្រោយ ។ compareTo() method អាចបំពេញនូវគោលបំនងនោះបាន ។ វាមានទំរង់ទូទៅដូចខាងក្រោម :

int compareTo(String *str*)

កម្មវិធីខាងក្រោមនេះបង្ហាញអំពីការតំរៀប strings នៅក្នុង array មួយ។ កម្មវិធីបានប្រើ compareTo() ដើម្បីកំនត់នូវលំដាប់តាម Bubble sort algorithm ។

បើសិនជាយើងចង់អោយវាមិនប្រកាន់អក្សរតូចធំនៅពេលធ្វើការប្រៀបធៀប strings ពីរនោះ គេត្រូវប្រើ :

int compareToIgnoreCase(String *str*)

method នេះអោយលទ្ធផលដូចនឹង compareTo() ដែរ គ្រាន់តែវាមិនប្រកាន់តួអក្សរតូចធំប៉ុណ្ណោះ ។

៩. ការរុករក String

String class បានផ្តល់ methods ពីរដែលអាចអោយយើងរក string ចំពោះតួអក្សរដែលយើងកំណត់ ឬ substring ណាមួយនោះ ។

- ✯ indexOf() រកតួអក្សរឬ substring ដែលកើតឡើងមុនគេបង្អស់ ។
- ✯ lastIndexOf() រកតួអក្សរឬ substring ដែលកើតឡើងខាងក្រោយគេបង្អស់ ។

នៅគ្រប់ករណីទាំងអស់ methods ទាំងនេះអោយតំលៃ index នៅត្រង់តួអក្សរ ឬ substring ដែលវាបានជួបប្រទះ ហើយបើវារកមិនឃើញទេនោះ វានឹងអោយតំលៃ –1 ។ វាមានទំរង់ជាច្រើនដូចខាងក្រោមនេះ :

int indexOf(int *ch*)

int lastIndexOf(int *ch*)

ក្នុងនេះ *ch* ជាតួអក្សរដែលគេចង់រកនៅក្នុង string ពីខាងដើម ឬពីខាងចុង ។

int indexOf(String *str*)
int lastIndexOf(String *str*)

ក្នុងនេះ *str* ជា substring ដែលគេចង់រកនៅក្នុង string ពីខាងដើម ឬពីខាងចុង ។

int indexOf(int *ch*, int *startIndex*)
int lastIndexOf(int *ch*, int *startIndex*)
int indexOf(String *str*, int *startIndex*)
int lastIndexOf(String *str*, int *startIndex*)

ក្នុងនេះ *startIndex* កំណត់អោយ index ដែលជាចំនុចចាប់ផ្តើមរុករក ។ ចំពោះ indexOf() ដំណើរការរកចាប់ផ្តើមពី *startIndex* រហូតដល់ខាងចុងនៃ string ។

រីង lastIndexOf() ដំណើរការរកដោយចាប់ផ្តើមពី *startIndex* រហូតមក index ត្រង់សូន្យ ។

កម្មវិធីខាងក្រោមបង្ហាញរបៀបប្រើ methods ផ្សេងៗ ខាងលើនេះដើម្បីរកតួអក្សរនៅក្នុង string ។

១០. ការផ្លាស់ប្តូរ String

ដោយសារ object នៃ String មានលក្ខណៈពិបាកផ្លាស់ប្តូរ នៅពេលដែលយើងចង់ធ្វើការកែប្រែនោះ ។ ពោលគឺយើងត្រូវតែចំលងវាទៅក្នុង StringBuffer ឬប្រើ methods មួយចំនួនដូចខាងក្រោមនេះដែលអាចអោយយើងធ្វើការកែប្រែបាន ។

១០.១ ការប្រើ substring()

គេអាចដំណើរការ substring ដោយប្រើ substring() method ។ វាមានពីរទំរង់ ។ ទំរង់ទីមួយគឺ:

String substring(int *startIndex*)

ក្នុងនេះ *startIndex* កំណត់អោយ index ដែល substring ត្រូវចាប់ផ្តើម ។ ទំរង់នេះអោយតំលៃនូវការចំលង substring ដែលចាប់ផ្តើមត្រង់ *startIndex* រហូតដល់ចុងបញ្ចប់នៃ string ។

ទំរង់ទីពីរនៃ substring() អាចអោយយើងកំណត់ទាំងការចាប់ផ្តើមនៃ substring និងចុងបញ្ចប់នៃ substring ។

String substring(int *startIndex*, int *endIndex*)

ក្នុងនេះ *startIndex* កំនត់អោយ index ចាប់ផ្តើម ហើយ *endIndex* កំនត់ត្រង់ចំនុចបញ្ឈប់ ។ វានឹងអោយតំលៃនូវតួអក្សរទាំងអស់ពី *startIndex* ដល់ *endIndex* តែមិនរួមបញ្ចូលនៅតួអក្សរត្រង់ចំនុចបញ្ឈប់ទេ។

កម្មវិធីខាងក្រោមនេះប្រើ substring() ដើម្បីជំនួសអោយគ្រប់ពាក្យដែលគេរកនៅក្នុង substring មួយ ។

១០.២ ការប្រើ concat()

គេអាចភ្ជាប់ strings ពីរដោយប្រើ concat() ។ វាមានទំរង់ដូចខាងក្រោម :

String concat(String *str*)

method នេះបង្កើត object ថ្មីមួយដែលបង្កពី string ជាមួយនឹងតំលៃរបស់ *str* ដោយបន្ថែមនៅខាងចុង string នោះ ។

ឧទាហរណ៍ : String s1 = "one";
String s2 = s1.concate("two");

បង្កបានជា string មានតំលៃ "onetwo" ទៅអោយ s2 ។

រ៉ាដូចគ្នាទៅនឹងការសរសេរតាមទំរង់ខាងក្រោមនេះដែរ : String s1 = "one";

String s2 = s1 + "two";

១០.៣ ការប្រើ replace()

replace() method ប្រើសំរាប់ជំនួសអោយតួអក្សរទាំងអស់នៅក្នុង String មួយដោយប្រភេទតួអក្សរមួយផ្សេងទៀត ។ ទំរង់ទូទៅរបស់វា គឺ:

String replace(char *original*, char *replacement*)

ក្នុងនេះ *original* ជាតួអក្សរដែលត្រូវជំនួសដោយតួអក្សរកំនត់ក្នុង *replacement* ។

ឧទាហរណ៍ : String s = "Hello".replace('l', 'w');

បង្កបានជា string មានតំលៃ “Hewwo” ទៅអោយ s ។

១០.៤ ការប្រើ trim()

trim() method ប្រើសំរាប់លុបបំបាត់ចោលដំណកឃ្លានៅខាងដើម string និងខាងចុង string ។ វាមានទំរង់ទូទៅដូចខាងក្រោម :

String *trim()*

ឧទាហរណ៍ : String s = " Hello World ".trim();

គេនឹងទទួលបាន string មានតំលៃ : “Hello World” ពី s ។

កម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រើ trim() method ។

១១. ការប្រើ valueOf() ដើម្បីបំលែងទិន្នន័យទៅជា String

valueOf() ប្រើសំរាប់បំលែងប្រភេទទិន្នន័យផ្សេងៗទៅជា String ។ វាជា overloaded method ។

```
static String valueOf(double num)
static String valueOf(long num)
static String valueOf(Object ob)
static String valueOf(char chars[])
```

១២. ការប្តូរតួអក្សរតូចធំនៅក្នុង String ២យ

toLowerCase() method ប្រើសំរាប់បំលែងតួអក្សរទាំងអស់នៅក្នុង string ពីអក្សរធំមកជាអក្សរតូច ។ រីង toUpperCase() method ប្រើសំរាប់បំលែងតួអក្សរទាំងអស់នៅក្នុង string ពីអក្សរតូចមកជាអក្សរធំ ។ វាមានទំរង់ ៖

String *toLowerCase()*
String *toUpperCase()*

methods ទាំងពីរនេះអោយតំលៃជា object នៃ String ដែលផ្ទុកអក្សរតូច ឬអក្សរធំ ។

កម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រើ toLowerCase() និង toUpperCase() ។

```
class ChangeCase{
    public static void main(String args[]){
        String s = "This is a test.";
        System.out.println("Original: " + s);
        String upper = s.toUpperCase();
        String lower = s.toLowerCase();
        System.out.println("Uppercase: " + upper);
        System.out.println("Lowercase: " + lower);
    }
}
```

១៣. អំពី StringBuffer

StringBuffer គឺជាពពួក class នៃ String ដែលផ្តល់នូវមុខងារនៃ strings ជាច្រើន។ ដូចយើងបានដឹងហើយថា String តាងអោយតួអក្សរមួយខ្សែមិនប្រែប្រួលនិងមានប្រវែងកំនត់។ ផ្ទុយទៅវិញ StringBuffer តាងអោយតួអក្សរមួយខ្សែដែលអាចសរសេរបាននិងអាចពង្រីកបាន។ StringBuffer អាចអោយគេបញ្ចូលតួអក្សរនិង substrings នៅផ្នែកកណ្តាល ឬបន្ថែមនៅខាងចុង string។ StringBuffer បង្កើនទំហំដោយស្វ័យប្រវត្ត ដើម្បីអោយគេអាចដាក់តួអក្សរបានគ្រប់គ្រាន់។

១៣.១ ទំរង់ constructors របស់ StringBuffer

StringBuffer មាន constructors ចំនួនបី :

StringBuffer()
StringBuffer(int *size*)
StringBuffer(String *str*)

ទំរង់ទីមួយជា default constructor (ពុំមានប៉ារ៉ាម៉ែត្រ) បំរុងទុកនូវទំហំ ១៦ តួអក្សរដោយមិនចាំបាច់បង្កើតទីតាំង។ ទំរង់ទីពីរមានការទទួលយកតំលៃជាចំនួនគត់មួយសំរាប់កំណត់ទំហំ buffer។ ទំរង់ទីបីមានការទទួលយក String មួយដែលកំណត់តំលៃដំបូងទៅអោយ object នៃ StringBuffer ព្រមទាំងបំរុងទុកនូវ ១៦ តួអក្សរទៀតដោយមិនបង្កើតទីតាំង។

១៣.២ ការប្រើ length() និង capacity()

ប្រសែង StringBuffer កំពុងប្រើអាចដឹងបានតាមរយៈ length() method ។ ហើយគេអាចដឹងចំណុះសរុបនៃការប្រើទីតាំងតាមរយៈ capacity() method ។ វាមានទំរង់ដូចខាងក្រោម :

int length()

int capacity()

កម្មវិធីខាងក្រោមនេះជាឧទាហរណ៍មួយបង្ហាញពីការប្រើ methods ទាំងនេះ។

```
// StringBuffer length vs. capacity.
class StringBufferDemo {
  public static void main(String args[]) {
    StringBuffer sb = new StringBuffer("Hello");
    System.out.println("buffer = " + sb);
    System.out.println("length = " + sb.length());
    System.out.println("capacity = " +
                                    sb.capacity());
  }
}
```

១៣.៣ ការប្រើ ensureCapacity()

បើសិនជាគេចង់ត្រៀមលក្ខណៈបង្កើតទីតាំងនូវទំហំមួយតាមចំនួនតួអក្សរ បន្ទាប់ពី StringBuffer បានបង្កើតឡើងនោះ គេអាចប្រើ ensureCapacity() ដើម្បីកំនត់ទំហំនៃ buffer ។ វាមានសារៈសំខាន់ បើយើងចង់ដឹងមុនថា គេនឹងបន្ថែមនូវ string តូចៗជាច្រើនទៅអោយ StringBuffer ។ វាមានទំរង់ដូចខាងក្រោមនេះ :

void ensureCapacity(int *capacity*)

ដែលក្នុងនេះ *capacity* បញ្ជាក់នូវទំហំនៃ buffer ។

១៣.៤ ការប្រើ setLength()

ដើម្បីកំណត់ប្រវែង buffer នៅក្នុង object នៃ StringBuffer គេប្រើ setLength() method ។ វាមានទំរង់ទូទៅដូចខាងក្រោម :

void setLength(int *len*)

ដែលក្នុងនេះ *len* កំណត់ប្រវែង buffer ។ តំលៃរបស់វាត្រូវតែវិជ្ជមាន ។

កាលណាយើងបង្កើនទំហំ buffer តួអក្សរ null ត្រូវបន្ថែមនៅខាងចុង buffer ដែលមានរួចហើយ ។ បើយើងហៅ

setLength() មកប្រើដោយមានតំលៃតូចជាងតំលៃកំពុងប្រើ ដែលបានមកពី length() នោះ តួអក្សរនឹងត្រូវបាត់បង់កាលណា វាបានផ្ទុកលើសពីប្រវែងដែលបានកំណត់ថ្មី ។

១៣.៥ ការប្រើ charAt() និង setCharAt()

តំលៃនៃតួអក្សរតែមួយអាចទទួលបានពី StringBuffer តាមរយៈ charAt() method ។ គេអាចកំណត់តំលៃនៃតួអក្សរមួយនៅក្នុង StringBuffer ដោយប្រើ setCharAt() ។ វាមាន :

```
char charAt(int where)
void setCharAt(int where, char ch)
```

ក្នុងទំរង់ទីមួយ *where* បញ្ជាក់អោយ index នៃតួអក្សរដែលគេនឹងទទួលបាន ។ រីឯទំរង់ទីពីរ *where* បញ្ជាក់អោយ index នៃតួអក្សរដែលត្រូវដាក់ ហើយ *ch* បញ្ជាក់នូវតំលៃថ្មីនៃតួអក្សរនោះ ។ *where* ត្រូវតែមានតំលៃវិជ្ជមាន និងមិនត្រូវកំនត់ទីតាំងអោយហួសខាងចុងបំផុតនៃ buffer ឡើយ ។

កម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រើ charAt() និង setCharAt() ។

១៣.៦ ការប្រើ getChars()

ដើម្បីចំលង substring មួយរបស់ StringBuffer ទៅដាក់ក្នុង array មួយ គេប្រើ getChars() method ។ វាមានទំរង់ទូទៅដូចខាងក្រោម :

void getChars(int *sourceStart*, int *sourceEnd*, char *target*[], int *targetStart*)

ក្នុងនេះ *sourceStart* បញ្ជាក់នូវ index នៃការចាប់ផ្តើមរបស់ substring បើយ *sourceEnd* បញ្ជាក់នូវ index នៃការបញ្ចប់តួអក្សរ(ប៉ុន្តែគេទទួលបានតួអក្សរចន្លោះពី *sourceStart* ទៅ *sourceEnd-1* ប៉ុណ្ណោះ) ។ array ដែលនឹងទទួលបាននូវ

តួអក្សរនោះត្រូវបានកំណត់ដោយ *target* ។ index នៅក្នុង *target* ដែល substring នឹងចំលងដាក់នោះគឺត្រូវប្រើតាម រយៈ *targetStart* ។ យើងត្រូវចាំថា array ឈ្មោះ *target* ត្រូវមានទំហំធំលួម ដើម្បីដាក់ចំនួនតួអក្សររបស់ substring បានគ្រប់គ្រាន់ ។

១៣.៧ ការប្រើ append()

append() method ប្រើសំរាប់ភ្ជាប់តំណាង string នៃប្រភេទទិន្នន័យណាមួយទៅខាងចុង object នៃ StringBuffer ។ វាមានទំរង់មួយចំនួនដូចខាងក្រោម :

StringBuffer append(String *str*)
StringBuffer append(int *num*)
StringBuffer append(Object *obj*)

ចូរសង្កេតនូវឧទាហរណ៍ខាងក្រោមនេះដើម្បីយល់បានពីការប្រើ append() ដែលគេអាចសរសេរតៗគ្នាច្រើនបាន ។

```
// Demonstrate append().
class appendDemo{
  public static void main(String args[]){
    String s;
    int a = 42;
    StringBuffer sb = new StringBuffer(40);
    s = sb.append("a = ").append(a).append("!").toString();
    System.out.println(s);
  }
}
```

១៣.៨ ការប្រើ insert()

insert() method ប្រើសំរាប់បញ្ចូល string មួយទៅក្នុង string មួយទៀត ។ វាមានចំនួនច្រើន ដែលអាចទទួលយកតំលៃនៃប្រភេទទិន្នន័យទំរង់ងាយទាំងអស់ថែមទាំងប្រភេទ String និង Object ។ ដូចទៅនឹង append() method ដែរ វាហៅ String.valueOf() មកប្រើ ដើម្បីទទួលបាននូវ string តំណាងអោយតំលៃដែលបានប្រើជាមួយនឹងវានោះ ។ ពេលនោះ string នេះត្រូវបានបញ្ចូលទៅក្នុង objects នៃ StringBuffer ។

វាមានទំរង់មួយចំនួនដូចខាងក្រោម :

StringBuffer insert(int *index*, String *str*)

StringBuffer insert(int *index*, char *ch*)

StringBuffer insert(int *index*, Object *obj*)

ក្នុងនេះ *index* បញ្ជាក់អោយ index នៅត្រង់ចំនុចដែលត្រូវ

បញ្ចូល string ទៅក្នុង objects នៃ StringBuffer ។

កម្មវិធីខាងក្រោមនេះបញ្ចូល "like" ទៅក្នុងចន្លោះ "I"

និង "Java" ។

```
// Demonstrate insert().
class insertDemo {
    public static void main(String args[]){
        StringBuffer sb = new StringBuffer("I Java!");
        sb.insert(2, "like ");
        System.out.println(sb);
    }
}
```

១៣.៩ ការប្រើ reverse()

គេអាចបញ្ច្រាសតួអក្សរនៅក្នុង object នៃ StringBuffer ដោយប្រើ reverse() method ។ វាមានទំរង់ដូចខាងក្រោម :

StringBuffer *reverse()*

កម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រើ reverse() ។

```
// Using reverse() to reverse a StringBuffer.
class ReverseDemo {
  public static void main(String args[]) {
    StringBuffer s = new StringBuffer("abcdef");
    System.out.println(s);
    s.reverse();
    System.out.println(s);
  }
}
```

១៣.១០ ការប្រើ delete() និង deleteCharAt()

StringBuffer អាចអោយគេលុបតួអក្សរដោយប្រើ delete() និង deleteCharAt() methods ។ Methods ទាំងនេះមានទំរង់ដូចខាងក្រោម:

StringBuffer delete(int *startIndex*, int *endIndex*)
StringBuffer deleteCharAt(int *loc*)

delete() លុបអក្សរមួយជួរចាប់ពី *startIndex* ទៅដល់ *endIndex* ចេញពី object នៃ StringBuffer ។ យើងត្រូវចាំថាតួអក្សរដែលត្រូវលុបនោះគឺចាប់ពី *startIndex* ទៅដល់ *endIndex–1* ប៉ុណ្ណោះ ។ វានឹងអោយលទ្ធផលជា object នៃ

StringBuffer ។

deleteCharAt() លុបតួអក្សរនៅត្រង់ index ដែលកំនត់ដោយ *loc* ។ វាអោយលទ្ធផលជា object នៃ StringBuffer ។ កម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រើ delete() និង deleteCharAt() methods ។

```
// Demonstrate delete() and deleteCharAt()
class deleteDemo {
    public static void main(String args[]) {
      StringBuffer sb = new StringBuffer(
                                "This is a test.");
      sb.delete(4, 7);
      System.out.println("After delete: " + sb);
      sb.deleteCharAt(0);
      System.out.println("After deleteCharAt: "
                                + sb);
    }
}
```

១៣.១១ ការប្រើ replace()

method ថ្មីមួយទៀតដែលបន្ថែមទៅអោយ StringBuffer នោះគឺ replace() ។ វាជំនួសអោយតួអក្សរមួយសំណុំដោយតួអក្សរមួយសំណុំផ្សេងទៀតនៅក្នុង object នៃ StringBuffer ។ ទំរង់ទូទៅរបស់វាគឺ :

StringBuffer replace(int *startIndex*,
int *endIndex*, String *str*)

substring ដែលត្រូវជំនួសនោះកំណត់ដោយ index ពី *startIndex* ដល់ *endIndex* ។ ហេតុនេះ substring

ទទួលបានគឺ ពី *startIndex* ដល់ *endIndex–1* ។ *str* ជា string ដែលត្រូវជំនួសទៅក្នុងចន្លោះ *startIndex* និង *endIndex* ។ វាអោយតំលៃជា object នៃ StringBuffer ។

កម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រើ replace() នេះ :

```
class replaceDemo {
    public static void main(String args[]) {
      StringBuffer sb = new StringBuffer("This is a test.");
      sb.replace(5, 7, "was");
      System.out.println("After replace: " + sb);
    }
}
```

១៣.១២ ការប្រើ substring()

គេអាចប្រើ substring() ជាមួយនឹង objects នៃ StringBuffer បានដែរ។ វាមានទំរង់ដូចខាងក្រោមនេះ :

String substring(int *startIndex*)

String substring(int *startIndex*, int *endIndex*)

ទំរង់ទីមួយអោយតំលៃជា substring ដែលចាប់ផ្តើមត្រង់ *startIndex* និងបន្តរហូតដល់ទីបញ្ចប់នៃ object របស់ StringBuffer។ ទំរង់ទីពីរអោយតំលៃជា substring ដែលចាប់ផ្តើមត្រង់ *startIndex* និងបន្តរហូតដល់ *endIndex–1* ។

១៤. អំពី StringTokenizer class

ដំណើរការអក្សរជារឿយៗមានការវិភាគទ្រង់ទ្រាយអក្សរដែលបានបញ្ចូល ។ ការវិភាគគឺបំបែកអក្សរទៅជាសំណុំបំណែកតូចៗដាច់ពីគ្នា ឬ tokens ដែលលំដាប់អក្សរអាចធ្វើអោយមានអត្ថន័យតាមចំណាងរបស់គេ ។ StringTokenizer class ផ្តល់នូវដំហានទីមួយក្នុងដំណើរការវិភាគនេះហៅថា *lexer* (lexical analyzer) ឬ *scanner* ។ StringTokenizer អនុវត្តតាម Enumeration interface ។ ហេតុនេះ អក្សរដែលគេបានបញ្ចូលនោះ គេអាចធ្វើការរាប់ tokens ដោយម្តងម្ភៃ

ពីគ្នាដែលមានក្នុងវាដោយប្រើ StringTokenizer ។

ដើម្បីប្រើ StringTokenizer គេត្រូវបញ្ជាក់អក្សរដែលត្រូវបញ្ចូល និងអក្សរដែលមាន delimiters (ដែនកំនត់) ។ delimiters គឺជាតួអក្សរដែលខ័ណ្ឌ tokens ។ តួអក្សរនិមួយៗនៅក្នុងអក្សរ delimiters ត្រូវគិតដែនកំណត់អោយបានត្រឹម ត្រូវ ឧទាហរណ៏ ",;:" កំណត់ delimiters នូវសញ្ញាក្បៀស សញ្ញាចុចក្បៀស និងសញ្ញាចុចពីរ ។ តាមធម្មតាការកំណត់ delimiters មានតួអក្សរដំណកឃ្លាដូចជា space, tab, newline និង carriage return ។

ទំរង់ constructors របស់ StringTokenizer គឺ :

StringTokenizer(String *str*)
StringTokenizer(String *str*, String *delimiters*)
StringTokenizer(String *str*, String *delimiters*, boolean *delimAsToken*)

ដែលក្នុងនេះ *str* គឺជា string ដែលត្រូវធ្វើការបំបែក ។ ទំរង់ constructor ទីមួយប្រើ delimiters ធម្មតា ។ ទំរង់ constructors ទីពីរនិងទីបីមានការកំណត់នូវ delimiters ។ ទំរង់ constructor ទីបី បើសិនជា *delimAsToken* មាន តំលៃ true នោះ delimiters ត្រូវអោយតំលៃជា tokens ដែរនៅពេល string ត្រូវបានវិភាគ ។

ផ្ទុយទៅវិញ delimiters មិនអោយតំលៃទេ ។ Delimiters មិនអោយតំលៃជា tokens ទេ នៅក្នុងទំរង់ constructors ទីមួយនិងទីពីរ ។

កាលណាគេបានបង្កើត object នៃ StringTokenizer នោះ method ឈ្មោះ nextToken() ត្រូវបានប្រើសំរាប់ទាញយក tokens ឥត្តា ។ hasMoreTokens() method អោយតំលៃ true កាលណាវានៅមាន tokens សំរាប់ទាញយកទៀត ។ ដោយសារ StringTokenizer អនុវត្តតាម Enumeration នោះ methods ឈ្មោះ

hasMoreElements() និង nextElement() ត្រូវបានប្រើផងដែរ ហើយវាធ្វើសកម្មភាពដូចទៅនឹង hasMoreTokens() និង nextToken() ។ StringTokenizer methods ត្រូវបានបង្ហាញក្នុងតារាងខាងក្រោម ។

នេះជាឧទាហរណ៍ដែលបង្កើតនូវ StringTokenizer សំរាប់ញែក "key=value" ជាគូ ។ សំណុំ នៃគូ "key=value" ឥឡូវត្រូវញែកដោយសញ្ញាក្បៀស ។

ឈ្មោះ Methods	ការអធិប្បាយ
int countTokens()	អោយចំនួន tokens ។
boolean hasMoreElements()	អោយតំលៃ true បើសិនជានៅមាន tokens មួយឬច្រើនក្នុង string និងអោយតំលៃ false កាលណាវាគ្មានទៀត ។
boolean hasMoreTokens()	អោយតំលៃ true បើសិនជានៅមាន tokens មួយឬច្រើនក្នុង string និងអោយតំលៃ false កាលណាវាគ្មានទៀត ។
Object nextElement()	អោយតំលៃ token នៅបន្ទាប់ជា Object មួយ ។

String nextToken() អោយតំលៃ token នៅបន្ទាប់ជា String

មួយ ។

String nextToken(String *delimiters*) អោយតំលៃ token

នៅបន្ទាប់ជា String មួយហើយកំណត់

delimiters string ទៅអោយ

delimiters នោះ ។

សំនួរឆ្លើយលំហាត់

១ - ចូរសរសេរកម្មវិធីមួយដែលអាចអោយគេប្រើ method របស់ String ឈ្មោះ compareTo ធ្វើការប្រៀបធៀប strings ពីរដែលបញ្ចូលដោយអ្នកប្រើប្រាស់ ។ លទ្ធផលរបស់វាអាចជា string ទីមួយមានតំលៃតូចជាង ឬ ដូចគ្នា ឬធំជាង string ទីពីរ ។

២ - ចូរសរសេរកម្មវិធីមួយដែលអាចអោយគេប្រើ method របស់ String ឈ្មោះ regionMatches ដើម្បី ធ្វើការប្រៀបធៀប strings ពីរដែលបញ្ចូលដោយអ្នកប្រើប្រាស់ ។ កម្មវិធីត្រូវតែបញ្ចូលចំនួនតួអក្សរ ដែលត្រូវធ្វើការប្រៀបធៀប ហើយចាប់ផ្តើមនៅត្រង់ index នៃការប្រៀបធៀប ។ កម្មវិធីត្រូវតែប្រាប់អោយដឹង រ៉ាតើ strings ដូចគ្នា ឬមិនដូចគ្នា ។ ហើយនៅពេលធ្វើការប្រៀបធៀប រ៉ាមិនប្រកាន់តួអក្សរតូចធំឡើយ ។

៣ - ចូរសរសេរកម្មវិធីមួយដែលអាចអោយគេបញ្ចូលអក្សរមួយឃ្លា ហើយពេលនោះលទ្ធផលរបស់វានឹងបង្ហាញអក្សរនោះពីរជួរ ដែលជួរទីមួយជាអក្សរធំទាំងអស់ និងជួរទីពីរជាអក្សរតូចទាំងអស់ ។

៤ - ចូរសរសេរកម្មវិធីមួយដែលអាចអោយគេបញ្ចូលតួអក្សរមួយឃ្លា រួចហើយបញ្ចូលតួអក្សរមួយតួដែលត្រូវរកនៅក្នុងអក្សរមួយឃ្លានោះដោយប្រើ method របស់ String ឈ្មោះ indexOf() ដើម្បីរកលេខត្រង់ទីតាំងនៃការកើតឡើងរបស់តួអក្សរមានក្នុងឃ្លានោះ ។

៥ - តើ trim() method របស់ String មានមុខងារសំរាប់ធ្វើអ្វី? ចូរសរសេរ code បង្ហាញពីការប្រើវា ។

៦ - តើ code ខាងក្រោមនេះនឹងអោយលទ្ធផលអ្វី កាលណាគេ compile និង run វា?

```
class RefEq {
    public static void main(String[] args) {
        String s = "ab" + "12";
        String t = "ab" + 12;
        String u = new String("ab12");
        System.out.println((s==t) + " " + (s==u));
    }
}
```