```java
<table>
<thead>
<tr>
<th>return type</th>
<th>method name</th>
<th>parameters / arguments / input</th>
</tr>
</thead>
<tbody>
<tr>
<td>(may be void)</td>
<td>replace</td>
<td>(char oldChar, char newChar)</td>
</tr>
</tbody>
</table>

To Do: Convert bitter to BIGGER

public class convertString {  // save as convertString.java/

    public static void main(String[] args) {

        String s1 = "bitter"
        String s2 = s1.replace(‘t’, ‘g’);
        String s3 = s2.toUpperCase();

        // print and skip to new line
        System.out.println("new String is: "+s3);

        // print and stay on same line.
        System.out.print(s3);
        System.out.println("done");

    }

}

Output:
new String is BIGGER
BIGGER done
Strings try to mimic primitive data types.

Most other classes require an import to be used.

Scanner - reads user input.

```java
import java.util.Scanner;

public class Input {

    public static void main(String[] args) {
        int x = 5;
        type name to value

        Scanner scan = new Scanner(System.in);
        type name value

        constructor - method that creates an object:

        System.out.println("Enter day");

        <Scanner API>
        which method to get a String from user? nextLine()

        String day = scan.nextLine();
        System.println("Happy " + day);
    }
}
```
To input date?

Suppose: <Show code>.

```java
int date = scan.nextInt();
String day = scan.nextLine();
```

User's input:

```
15 "In" day
Wednesday
```

Ideas to fix?

Use another `nextLine()` after `nextInt()`.

```java
int year = scan.nextInt();
scan.nextLine(); // reading + ignoring "In"
String day = scan.nextLine();
```
Different data types lead to need to convert data types.

Data Type conversion

Narrowing conversion - information is lost (sometimes not allowed by Java i.e. won't compile)

Ex: Storing double (8 bytes) as int (4 bytes)
    "int (4"
    "char(2"

Widening conversion - no info lost

Ex: int as double
    char as int

How to convert?

Assignment: assign variable of one type to another

Ex: double x;
    int y = 10;

    x = y;  // x = 10.0 storing int (10) as double (10.0) (widening)

    x = 5.0
    y = x;  // y = 5 storing 5.0 to 5 would not compile

Casting - Explicitly convert variable

y = (int) x  => y = 5
Last way: Promotion through operations

double q = x/y; q = 0.5 (as expected)
int q2 = (int)x/y; q2 = 0 (rounds down to nearest int)

x = 5;

int q2 = (int)x/y; // can't store 0.5 as int (narrowing)
// so add cast

double q = x/y; // q = 0.5