STRING REVIEW: STARTSWITH()

public boolean beginsHow(String str)

Given a String, return **true** if it starts with "How". **false** if it doesn't.
The method is **case-sensitive**. Capitals DON'T match their lowercase counterparts, i.e. "Abc" does not match "aBC".

Explanation:
Begin completing this method as usual:

```
public boolean beginsHow(String str) {
  boolean begins = false;
  if (_____) {
    begins = true;
  }
  return begins;
}
```

Use the String method **startsWith(String s)** inside the parentheses:

```
public boolean beginsHow(String str) {
  boolean begins = false;
  if ( str.startsWith("How") ) {
    begins = true;
  }
  return begins;
}
```

The method below uses the other version of **startsWith**(String s, **int start**),
that has a 2nd parameter indicating at which **position** to start looking.

```
public boolean beginsHow(String str) {
  boolean begins = false;
  if ( str.startsWith("How",0) ) {
    begins = true;
  }
  return begins;
}
```

In the code above, the **0** in startsWith() starts the search
at the beginning of the string, with the 1st character.
So **startsWith("abc")** and **startsWith("abc",0)** do the same thing.

____

You can also start the search at other positions:
1 means start at the **2nd** character
2 means start at the **3rd** character

```
public boolean beginsHowAtIndex1(String str) {
  boolean beginsAtPos1 = false;
  if ( str.startsWith("How", 1) ) {
    beginsAtPos1 = true;
  }
  return beginsAtPos1;
}
```

The method above will match strings like
"**x**How are you?" or "**#**How is that possible?"

____

STRING REVIEW: STARTSWITH()

Note: you can also use an expression for where to start:

int **len** = str.length();
int **mid** = str.length()/2;
len-1 means start at the last character
len-2 means start at the 2nd-to-last character
len-3 means start at the 3rd-to-last character
mid means start at the middle character*
*If the string has an even length, there are two middle characters
and **mid** is the position of the 2nd one.

For example:
str.startsWith("mm", mid) matches "reco**mm**end"
str.startsWith("mm", mid-1) matches "su**mm**er"
str.startsWith("ed", len-2) matches "trust**ed**" [an alternative to **endsWith()**!]

Finally, note that -- unlike with **substring(int start, int stop)**,
which has restrictions on the range of values for its **start** and **stop** parameters --
there are **NO RESTRICTIONS** on the range of values for the **start** parameter of **startsWith()**.
That is, a negative nunber or one larger than the length of the string will **NOT** cause a runtime error.

This **internal** error checking means that your programs can often be much easier to write
using **startsWith()** than those that use combinations of **length()**, **substring()** and/or **equals()**.

---

public boolean endsGerund(String str)

Given a String, return **true** if it's a **gerund**, i.e. it **ends with "ing"**. Return **false** if it doesn't.
The method is **case-sensitive**. Capitals matter DON'T match their lowercase counterparts, i.e. "Abc" does not match "aBC".

Explanation:
Begin completing this method as usual:

```
public boolean endsGerund(String str) {
  boolean ends= false;
  if (_____) {
    ends = true;
  }
  return ends;
}
```

STRING REVIEW: STARTSWITH()

Use the String method **endsWith()** inside the parentheses:

```
public boolean endsGerund(String str) {
  boolean ends = false;
  if ( str.endsWith("ing") ) {
    ends = true;
  }
  return ends;
}
```

Note that - unlike **startsWith()** - there are **NOT** two versions of **endsWith()**.
That is, there is no version with a 2nd parameter.
That means that any string you test will only return true if it
actually **ENDS WITH** the string you are testing for.

NOTE: There is actually a way to solve this problem using **startsWith()**.

```
public boolean endsGerund(String str) {
  boolean ends = false;
  int len = str.length();
  if ( str.startsWith("ing",len-3) ) {
    ends = true;
  }
  return ends;
}
```

Why do we use "len**-3**" for the 2nd parameter?
Because the length of **"ing"** is **3**.
This causes **startsWith()** to start checking beginning at the 3rd-to-last character.

If we wanted to check whether a String ends with **"ment"**,
we would use **len-4**: **str.startsWith("ment",len-4)**
This causes **startsWith()** to start checking beginning at the 4th-to-last character.

If we wanted to check whether a String ends with **"ed"**,
we would use **len-2**: **str.startsWith("ed",len-2)**
This causes **startsWith()** to start checking beginning at the 2nd-to-last character.

public boolean beginsHowIgnoreCase(String str)

Given a String, return **true** if it starts with "How". **false** if it doesn't.
The method is **case-INSENSITIVE**, that is, capitals (upper case letters) and lower case letters **match**.

Explanation:
This can be done with just two easy changes:

```
public boolean beginsHowIgnoreCase(String str) {
```

STRING REVIEW: STARTSWITH()

```
  boolean begins = false;
  str = str.toLowerCase();
  if (str.startsWith("how",0)) {
    begins = true;
  }
  return begins;
}
```

(1) Use **toLowerCase()**, which creates a version of the string where all letters are lower case.
(2) Change the 1st parameter in **startsWith()** from "How" to **"how"** (all lowercase).

Note that you need to **RE-ASSIGN** the value of this new string to the **str** variable:

str = str.toLowerCase();

---

public boolean endsGerundIgnoreCase(String str)

Given a String, return **true** if it **ends with "ing"**. **false** if it doesn't.
The method is **case-INSENSITIVE**, that is, capitals (upper case letters) and lower case letters **match**.

Explanation:
This can be done with just one easy change:

```
public boolean endsGerundIgnoreCase(String str) {
  boolean ends = false;
  str = str.toLowerCase();
  if ( str.endsWith("ing") ) {
    ends = true;
  }
  return ends;
}
```

(*) Use **toLowerCase()**, which creates a version of the string where all letters are lower case.

Note that you need to **RE-ASSIGN** the value of this new string to the **str** variable:

str = str.toLowerCase();

---

ublic boolean almostEndsFUL(String str)

Given a String, return **true** if it **ALMOST** ends with the suffix **"ful"**.
Return **false** if it doesn't.

STRING REVIEW: STARTSWITH()

What does **ALMOST** mean exactly?
In this case, if you were to chop off the **last character**,
return true if that truncated substring ends with "ful".

Explanation: We've already mentioned that **endsWith()** does **NOT** have two versions.
That is, there is no version with a 2nd parameter, as with **startsWith()**.
That means that any string you test will only return true if it
actually **ENDS WITH** the string you are testing for.

That being said, there is a **WORKAROUND** if you want to test whether
a string "ends with" a letter sequence one or more characters **BEFORE**
the end of the string. That is, by ignoring one or more characters at the end.
This can be done using **startsWith()**!.

Begin by writing a method to test whether a string ends with "ful".

```
public boolean almostEndsFUL(String str) {
  boolean ends = false;
  int len = str.length();
  if (str.startsWith("ful",len-3)) {
    ends = true;
  }
  return ends;
}
```

Change "len-**3**" to "len-**4**" !
This will direct startsWith() to begin checking, not at the 3rd-to-last letter,
but beginning at the 4th-to-last letter from the end.

```
public boolean almostEndsFUL(String str) {
  boolean ends = false;
  int len = str.length();
  if (str.startsWith("ful",len-4)) {
    ends = true;
  }
  return ends;
}
```

```
public boolean almostEndsFUL(String str) {
  int len = str.length();
  boolean answer = str.startsWith("ful",len-4);
  return answer;
}

public boolean almostEndsFUL2(String str) {
  int len = str.length();
  String s = str.substring(0,len-1);
```

STRING REVIEW: STARTSWITH()

```
  boolean answer = s.endsWith("ful");
  return answer;
}
```