See the Assessment Guide for information on how to interpret this report.

ASSESSMENT SUMMARY

```
Compilation:  PASSED
API:          PASSED

SpotBugs:     PASSED
PMD:          PASSED
Checkstyle:   FAILED (0 errors, 6 warnings)

Correctness:  40/40 tests passed
Memory:       No tests available for autograding.
Timing:       No tests available for autograding.

Aggregate score: 100.00%
[ Compilation: 5%, API: 5%, Style: 0%, Correctness: 90% ]
```

ASSESSMENT DETAILS

```
The following files were submitted:
----------------------------------
 147 Feb 21 18:53 6-by-5.png
 134 Feb 21 18:53 AnnotationType.java
147K Feb 21 18:53 COS_126.xml
142K Feb 21 18:53 COS_126.xml.2020.1
1.9K Feb 21 18:53 CREDITS
 189 Feb 21 18:53 Class.java
 850 Feb 21 18:53 Computer\ Science.iml
 128 Feb 21 18:53 Enum.java
 259 Feb 21 18:53 File\ Header.java
2.0K Feb 21 18:53 Huntingtons.class
1.7K Feb 21 18:53 Huntingtons.java
 133 Feb 21 18:53 Interface.java
3.8K Feb 21 18:53 KernelFilter.class
5.3K Feb 21 18:53 KernelFilter.java
4.2K Feb 21 18:53 Project.xml
102K Feb 21 18:53 baboon-blue.png
129K Feb 21 18:53 baboon-gray.png
 99K Feb 21 18:53 baboon-green.png
101K Feb 21 18:53 baboon-red.png
234K Feb 21 18:53 baboon.png
1.1K Feb 21 18:53 checkstyle-idea.xml
 10M Feb 21 18:53 chromosome4-hd.txt
 10M Feb 21 18:53 chromosome4-healthy.txt
 15K Feb 21 18:53 codeInsightSettings.xml
 142 Feb 21 18:53 codeStyleConfig.xml
 384 Feb 21 18:53 compiler.xml
 90K Feb 21 18:53 earth-gray.png
118K Feb 21 18:53 earth.png
 201 Feb 21 18:53 encodings.xml
 267 Feb 21 18:53 externalDependencies.xml
283K Feb 21 18:53 f16-gray.png
415K Feb 21 18:53 f16.png
 290 Feb 21 18:53 file.template.settings.xml
 352 Feb 21 18:53 findbugs-idea.xml
190K Feb 21 18:53 fishingboat-gray.png
214K Feb 21 18:53 fishingboat.png
 560 Feb 21 18:53 introcs.xml
 190 Feb 21 18:53 lift.xml
 14K Feb 21 18:53 logo.png
 215 Feb 21 18:53 misc.xml
  58 Feb 21 18:53 module-info.java
 273 Feb 21 18:53 modules.xml
 102 Feb 21 18:53 package-info.java
 69K Feb 21 18:53 penguins-gray.png
130K Feb 21 18:53 penguins.png
342K Feb 21 18:53 peppers-gray.png
495K Feb 21 18:53 peppers.png
245K Feb 21 18:53 pipe-gray.png
345K Feb 21 18:53 pipe.png
 173 Feb 21 18:53 profiles_settings.xml
  90 Feb 21 18:53 repeats0.txt
 188 Feb 21 18:53 repeats10.txt
  37 Feb 21 18:53 repeats12.txt
 739 Feb 21 18:53 repeats180.txt
 742 Feb 21 18:53 repeats181.txt
  43 Feb 21 18:53 repeats2.txt
 305 Feb 21 18:53 repeats35.txt
```

```
  305 Feb 21 18:53 repeats36.txt
  304 Feb 21 18:53 repeats39.txt
  121 Feb 21 18:53 repeats4.txt
  305 Feb 21 18:53 repeats40.txt
  325 Feb 21 18:53 repeats64.txt
  185 Feb 21 18:53 repeats9.txt
  357 Feb 21 18:53 saveactions_settings.xml
  39K Feb 21 18:53 shield-gray.png
  36K Feb 21 18:53 shield.png
  20K Feb 21 18:53 sunflowers-gray.png
  37K Feb 21 18:53 sunflowers.png
 6.4K Feb 21 18:53 workspace.xml


********************************************************************************
*  COMPILING
********************************************************************************


% javac Huntingtons.java
*-----------------------------------------------------------

% javac KernelFilter.java
*-----------------------------------------------------------



================================================================


Checking the APIs of your programs.
*-----------------------------------------------------------
Huntingtons:

KernelFilter:

================================================================


********************************************************************************
*  CHECKING STYLE AND COMMON BUG PATTERNS
********************************************************************************


% spotbugs *.class
*-----------------------------------------------------------


================================================================


% pmd .
*-----------------------------------------------------------


================================================================


% checkstyle *.java
*-----------------------------------------------------------
[WARN] KernelFilter.java:24:21: The local variable 'R' must start with a lowercase letter and use camelCase. [LocalVariableName]
[WARN] KernelFilter.java:26:21: The local variable 'B' must start with a lowercase letter and use camelCase. [LocalVariableName]
[WARN] KernelFilter.java:127:21: The local variable 'R' must start with a lowercase letter and use camelCase. [LocalVariableName]
[WARN] KernelFilter.java:129:21: The local variable 'B' must start with a lowercase letter and use camelCase. [LocalVariableName]
Checkstyle ends with 0 errors and 4 warnings.

% custom checkstyle checks for Huntingtons.java
*-----------------------------------------------------------

% custom checkstyle checks for KernelFilter.java
*-----------------------------------------------------------
[WARN] KernelFilter.java:116:30: '-4' looks like an unnecessary constant. [MagicNumber]
[WARN] KernelFilter.java:117:34: '-4' looks like an unnecessary constant. [MagicNumber]
Checkstyle ends with 0 errors and 2 warnings.


================================================================


********************************************************************************
*  TESTING CORRECTNESS
********************************************************************************

Testing correctness of Huntingtons
*-----------------------------------------------------------
Running 10 total tests.

Test 1: check output format of main() for inputs from assignment specification
  % java-introcs Huntingtons repeats4.txt
  max repeats = 4
  not human
```

```
    % java-introcs Huntingtons repeats64.txt
    max repeats = 64
    Huntington’s

    % java-introcs Huntingtons chromosome4-hd.txt
    max repeats = 79
    Huntington’s

    % java-introcs Huntingtons chromosome4-healthy.txt
    max repeats = 19
    normal

  ==> passed

  Test 2: check correctness of main() for inputs from assignment specification
    % java-introcs Huntingtons repeats4.txt
    % java-introcs Huntingtons repeats64.txt
    % java-introcs Huntingtons chromosome4-hd.txt
    % java-introcs Huntingtons chromosome4-healthy.txt
  ==> passed

  Test 3: check maxRepeats() for DNA from files (with whitespace removed)
    * file = repeats0.txt
    * file = repeats2.txt
    * file = repeats4.txt
    * file = repeats9.txt
    * file = repeats10.txt
    * file = repeats12.txt
    * file = repeats35.txt
    * file = repeats36.txt
    * file = repeats39.txt
    * file = repeats40.txt
    * file = repeats64.txt
    * file = repeats180.txt
    * file = repeats181.txt
  ==> passed

  Test 4: check maxRepeats() for DNA from files (with whitespace removed)
    * file = chromosome4-hd.txt
    * file = chromosome4-healthy.txt
  ==> passed

  Test 5: check maxRepeats() for random DNA of length n
    * 10000 random strings of length 10
    * 10000 random strings of length 20
    * 10000 random strings of length 30
    * 10000 random strings of length 100
    * 10000 random strings of length 200
    * 10000 random strings of length 500
  ==> passed

  Test 6: check removeWhitespace() for inputs from files
    * file = repeats0.txt
    * file = repeats2.txt
    * file = repeats4.txt
    * file = repeats9.txt
    * file = repeats10.txt
    * file = repeats12.txt
    * file = repeats35.txt
    * file = repeats36.txt
    * file = repeats39.txt
    * file = repeats40.txt
    * file = repeats64.txt
    * file = repeats180.txt
    * file = repeats181.txt
  ==> passed

  Test 7: check removeWhitespace() for DNA from files
    * file = chromosome4-hd.txt
    * file = chromosome4-healthy.txt
  ==> passed

  Test 8: check maxRepeats() for random DNA of length n
    * 10000 random strings of length 10 over alphabet { 'A', 'C', 'G', 'T' }
    * 10000 random strings of length 10 over alphabet { 'A', 'C', 'G', 'T', ' ' }
    * 10000 random strings of length 10 over alphabet { 'A', 'C', 'G', 'T', ' ', '\n' }
    * 10000 random strings of length 10 over alphabet { 'A', 'C', 'G', 'T', ' ', '\n', '\t' }
    * 10000 random strings of length 20 over alphabet { 'A', 'C', 'G', 'T' }
    * 10000 random strings of length 20 over alphabet { 'A', 'C', 'G', 'T', ' ' }
    * 10000 random strings of length 20 over alphabet { 'A', 'C', 'G', 'T', ' ', '\n' }
    * 10000 random strings of length 20 over alphabet { 'A', 'C', 'G', 'T', ' ', '\n', '\t' }
    * 10000 random strings of length 100 over alphabet { 'A', 'C', 'G', 'T' }
    * 10000 random strings of length 100 over alphabet { 'A', 'C', 'G', 'T', ' ' }
    * 10000 random strings of length 100 over alphabet { 'A', 'C', 'G', 'T', ' ', '\n' }
    * 10000 random strings of length 100 over alphabet { 'A', 'C', 'G', 'T', ' ', '\n', '\t' }
  ==> passed

  Test 9: check diagnose() for given value of maxRepeats
    * maxRepeats = 0
    * maxRepeats = 9
    * maxRepeats = 10
    * maxRepeats = 35
    * maxRepeats = 36
    * maxRepeats = 39
    * maxRepeats = 40
    * maxRepeats = 180
    * maxRepeats = 181
  ==> passed

  Test 10: check diagnose() for range of values of maxRepeats
    * 0 to 9
    * 10 to 35
    * 36 to 39
    * 40 to 180
    * 180 to 1000
  ==> passed


  Huntingtons Total: 10/10 tests passed!


  ================================================================
  Testing correctness of KernelFilter
  *-----------------------------------------------------------
  Running 30 total tests.

  Test 1: check correctness of identity() for given grayscale PNG files
    * 6-by-5.png
    * baboon-gray.png
    * sunflowers-gray.png
    * earth-gray.png
    * penguins-gray.png
  ==> passed

  Test 2: check correctness of identity() for given color PNG files
    * baboon.png
    * baboon-red.png
    * baboon-green.png
    * baboon-blue.png
    * sunflowers.png
    * earth.png
    * penguins.png
  ==> passed

  Test 3: check correctness of identity() for random grayscale pictures
    * 1000 random 9-by-9 grayscale images
    * 1000 random 5-by-8 grayscale images
    * 1000 random 7-by-6 grayscale images
    * 1000 random 1-by-8 grayscale images
    * 1000 random 8-by-1 grayscale images
    * 1000 random 1-by-1 grayscale images
  ==> passed

  Test 4: check correctness of identity() for random color pictures
    * 1000 random 10-by-10 color images
    * 1000 random 12-by-17 color images
    * 1000 random 16-by-13 color images
  ==> passed

  Test 5: check correctness of gaussian() for given grayscale PNG files
    * 6-by-5.png
    * baboon-gray.png
    * sunflowers-gray.png
    * earth-gray.png
    * penguins-gray.png
  ==> passed

  Test 6: check correctness of gaussian() for given color PNG files
    * baboon.png
    * baboon-red.png
    * baboon-green.png
    * baboon-blue.png
    * sunflowers.png
    * earth.png
    * penguins.png
  ==> passed

  Test 7: check correctness of gaussian() for random grayscale pictures
    * 1000 random 9-by-9 grayscale images
    * 1000 random 5-by-8 grayscale images
    * 1000 random 7-by-6 grayscale images
    * 1000 random 1-by-8 grayscale images
    * 1000 random 8-by-1 grayscale images
    * 1000 random 1-by-1 grayscale images
  ==> passed

  Test 8: check correctness of gaussian() for random color pictures
    * 1000 random 10-by-10 color images
    * 1000 random 12-by-17 color images
    * 1000 random 16-by-13 color images
  ==> passed

  Test 9: check correctness of sharpen() for given grayscale PNG files
    * 6-by-5.png
    * baboon-gray.png
    * sunflowers-gray.png
    * earth-gray.png
    * penguins-gray.png
  ==> passed

  Test 10: check correctness of sharpen() for given color PNG files
    * baboon.png
    * baboon-red.png
    * baboon-green.png
    * baboon-blue.png
    * sunflowers.png
    * earth.png
    * penguins.png
  ==> passed

  Test 11: check correctness of sharpen() for random grayscale pictures
    * 1000 random 9-by-9 grayscale images
    * 1000 random 5-by-8 grayscale images
    * 1000 random 7-by-6 grayscale images
    * 1000 random 1-by-8 grayscale images
    * 1000 random 8-by-1 grayscale images
    * 1000 random 1-by-1 grayscale images
  ==> passed

  Test 12: check correctness of sharpen() for random color pictures
    * 1000 random 10-by-10 color images
    * 1000 random 12-by-17 color images
    * 1000 random 16-by-13 color images
  ==> passed

  Test 13: check correctness of laplacian() for given grayscale PNG files
    * 6-by-5.png
    * baboon-gray.png
    * sunflowers-gray.png
    * earth-gray.png
    * penguins-gray.png
  ==> passed

  Test 14: check correctness of laplacian() for given color PNG files
    * baboon.png
    * baboon-red.png
    * baboon-green.png
    * baboon-blue.png
    * sunflowers.png
    * earth.png
    * penguins.png
  ==> passed

  Test 15: check correctness of laplacian() for random grayscale pictures
    * 1000 random 9-by-9 grayscale images
    * 1000 random 5-by-8 grayscale images
    * 1000 random 7-by-6 grayscale images
    * 1000 random 1-by-8 grayscale images
    * 1000 random 8-by-1 grayscale images
    * 1000 random 1-by-1 grayscale images
  ==> passed

  Test 16: check correctness of laplacian() for random color pictures
    * 1000 random 10-by-10 color images
    * 1000 random 12-by-17 color images
    * 1000 random 16-by-13 color images
  ==> passed

  Test 17: check correctness of emboss() for given grayscale PNG files
    * 6-by-5.png
    * baboon-gray.png
    * sunflowers-gray.png
    * earth-gray.png
    * penguins-gray.png
  ==> passed

  Test 18: check correctness of emboss() for given color PNG files
    * baboon.png
    * baboon-red.png
    * baboon-green.png
    * baboon-blue.png
    * sunflowers.png
    * earth.png
    * penguins.png
  ==> passed

  Test 19: check correctness of emboss() for random grayscale pictures
    * 1000 random 9-by-9 grayscale images
    * 1000 random 5-by-8 grayscale images
    * 1000 random 7-by-6 grayscale images
    * 1000 random 1-by-8 grayscale images
    * 1000 random 8-by-1 grayscale images
    * 1000 random 1-by-1 grayscale images
  ==> passed

  Test 20: check correctness of emboss() for random color pictures
    * 1000 random 10-by-10 color images
    * 1000 random 12-by-17 color images
    * 1000 random 16-by-13 color images
  ==> passed

  Test 21: check correctness of motionBlur() for given grayscale PNG files
    * 6-by-5.png
    * baboon-gray.png
    * sunflowers-gray.png
    * earth-gray.png
    * penguins-gray.png
  ==> passed

  Test 22: check correctness of motionBlur() for given color PNG files
    * baboon.png
    * baboon-red.png
    * baboon-green.png
    * baboon-blue.png
    * sunflowers.png
    * earth.png
    * penguins.png
  ==> passed

  Test 23: check correctness of motionBlur() for random grayscale pictures
    * 1000 random 9-by-9 grayscale images
    * 1000 random 5-by-8 grayscale images
    * 1000 random 7-by-6 grayscale images
    * 1000 random 1-by-8 grayscale images
    * 1000 random 8-by-1 grayscale images
    * 1000 random 1-by-1 grayscale images
  ==> passed

  Test 24: check correctness of motionBlur() for random color pictures
    * 1000 random 10-by-10 color images
    * 1000 random 12-by-17 color images
    * 1000 random 16-by-13 color images
  ==> passed

  Test 25: check that identity() does not mutate Picture argument
    * baboon.png
    * sunflowers.png
    * earth.png
    * penguins.png
  ==> passed

  Test 26: check that gaussian() does not mutate Picture argument
    * baboon.png
    * sunflowers.png
    * earth.png
    * penguins.png
  ==> passed

  Test 27: check that sharpen() does not mutate Picture argument
    * baboon.png
    * sunflowers.png
    * earth.png
    * penguins.png
  ==> passed

  Test 28: check that laplacian() does not mutate Picture argument
    * baboon.png
    * sunflowers.png
    * earth.png
    * penguins.png
  ==> passed

  Test 29: check that emboss() does not mutate Picture argument
    * baboon.png
    * sunflowers.png
    * earth.png
    * penguins.png
  ==> passed

  Test 30: check that motionBlur() does not mutate Picture argument
    * baboon.png
    * sunflowers.png
    * earth.png
    * penguins.png
  ==> passed


  KernelFilter Total: 30/30 tests passed!
```

```
=================================================================
```