A. The Country Class

Country
protected int cNumber,cPopulation protected String cName protected double cGNI, cPCI, cStandard protected int cPopulation
public Country() public void modifyMe(Country thisCountry) public void inputData(int x) public String printMe() public double getPCI() public String getCountry() public String setPCI() protected void finalize() throws java.lang.Throwable protected void destroyMe(Object thisObj)

A1. The Country() Constructor

```
START
    cNumber, cGNI, cPCI, cPopulation = 0;
    cName = “”;
STOP
```

A2. The modifyMe(Country thisCountry) Method

```
START
    cNumber = thisCountry.cNumber;
    cName = thisCountry.cName;
    cGNI = thisCountry.cGNI;
    cPCI = thisCountry.cPCI;
    cPopulation = thisCountry.cPopulation;
STOP
```

A3. The inputData(int x) Method

```
START;=
    Let countryHeading and cNumberString be strings;
    Prompt for countryHeading and cNumberString;
    cNumber = cNumberString;
    Prompt for cNumber, cName, cGNI, and cPopulation;
    cPCI = cGNI/cPopulation;
STOP
```

A4. The printMe() Method: Returns a string

```
START
    Let printString be a string
    printString = "Country Number: " + cNumber + "\n" + "Name: " + cName + "\n" +
```

"Gross National Income: " + cGNI + "\n" + "Population: " + cPopulation + "\n" + "Per Capita Income: " + cPCI + "\n" + "Standard Deviation: " + cStandard;
Return printString;
STOP

A5. The getCountry() Method: Returns an Integer
START
Return cNumber;
STOP

A6. the getPCI() method: returns a double
START
Return cPCI;
STOP

A7. The setPCI(double thisPCI) Method: returns a double
START
cPCI = this.PCI;
Return thisPCI;
STOP

A8. The finalize Method
START
destroyMe(this);
STOP

A9. The destroyMe(thisObj) Method
START
thisObj = null;
System.gc();
STOP

B. The SortSimulator Class

SortSimulator
Public static Country[] countriesList Public static String inputString Public static final String countrySorterHEAD = “Country Sorter of Petr Bowles” Public static final int DEFAULT_NUMBER = 0 Static FileOutputStream CountryFileWriter Static ObjectOutputStram CountryObjectWriter Static final String CountryFileString = “JavaDataFiles” + File.separator + “Country.dat” Static FileInputStream CountryFileReader Static ObjectInputStream CountryObjectReader Static Vector <Country> countryVector Static double noCompTime = 0
public static void main(String[] args) Public static void inputCountries() Public static void listCountries(Country[] thisList, double compTime)

```
Public static void mergeSortController(Country[] thisList)
Public static Country[] mergeSortCountries(Country[] thisList, int first, int last)
Public static Country[] mergeLeftRight(Country[] thisList, Country[] leftList, Country[]
rightList)
Public static void bubbleSort(Country[] thisList)
Public static void exchangeSelection(Country[] thisList)
Public static void straightSelection(country[] thisList)
Public static void quickSortController(Country[] thisList)
Public static Country[] quickSortCountries(Country[] thisList, int first, int last)
Public static Country [] vToArray(Vector <Country> thisVector)
Public static Country[] loadCountries() throws Exception, IOException
Public static void createFile()
Public static void deleteFile()
```

B1. The main(String[] args) Method

START

Declare String promptString to a string containing all menu options
Declare boolean exitTime = false
Declare int userOption, i to integers
Declare copyList to a new Country array equal to countriesList
While (exitTime is false) do the following:
If (countryVector is not null) call vToArray for countriesList
Take input for inputString
Set userOption to the inputString
Call a switch for userOption with the following options:
Case 1: cal;l inputCountries
Case 2: if countriesList has a length, call listCountries
Case 3: if countriesList has a length, populate copyList with countriesList, call mergeSortController for copyList
Case 4:if countriesList has a length, populate copyList with countriesList, call bubbleSort for copyList
Case 5:if countriesList has a length, populate copyList with countriesList, call quickSortController for copyList
Case 6:if countriesList has a length, populate copyList with countriesList, call exchangeSelection for copyList
Case 7:if countriesList has a length, populate copyList with countriesList, call straightSelection for copyList
Case 8: call createFile
Case 9: call deleteFile
Case 0: set exitTime to true
End while

STOP

B2. The inputCountries() Method

START

Declare loadSize, x , and cLimit to integers

Declare thisList to a new Country array set to null
Call loadCountries for thisList with try catch surrounding it
If thisList is not null set loadSize to thisList length
Else loadsize is 0
Declare countryVector to a new vector
Call ensureCapacity of loadSize for countryVector
for(x = 1 to loadSize incrementing by 1) do the following
Add thisList[x-1] to position x -1 of countryVector
End for
Set cLimit to record user input
Ensure capacity of countryVector for loadSize + cLimit
for (x = 1 to cLimit incrementing by 1) do the following
Set dummy to a new country
Call inputData for position x of dummy
Add dummy to countryVector
End for
Set thisList to a new Country array of size of countryVector
Call vToArray for thisList using countryVector
Try
Set CountryFileWriter to a new FileOutputStream for CountryFileString
Set CountryObjectWriter to new objectOutputStream for CountryFileWriter
Call CountryObjectWriter.writeObject for thisList
Call CountryObjectWriter flush
Call CountryObjectWriter close
Catch Ex1 and Ex2 with message dialogs
STOP

B3. The listCountries(Country[] thisList, double compTime) Method
START
Declare String outString equal to “Members of the list are:”
Declare string compString = Computation time is: “
Declare x as an Integer
For (x = 1 to thisList.length incrementing by 1) do the following
Add thisList[x-1] to outString
End for
Call output dialog for outString
STOP

B4. The mergeSortController(Country[] thisList) Method
START
Declare startTime, endTime and compTime as doubles
Call system.nanoTime for startTime
Try
Set thisList equal to loadCountries
Catch
Exception ex

Set thisList equal to mergeSortCountries
Call system.nanoTime for endTime
Set compTime to the difference between endTime and startTime
Call listCountries with thisList and compTime
STOP

B5. The mergeSortCountries(Country[] thisList, int first, int last) Method
START
Declare k, l, and middle as integers
Set middle to first plus last divided by 2
If (first < last) do the following
Declare country array thisLeft of size middle - first +1
for(k = 0 to middle incrementing by 1) do the following
thisLeft of position K equals new country
Populate thisLeft position K with thisList position K
End for
Call mergerSortCountries for thisLeft using thisLeft, 0, and Middle
Declare country array thisRight of size middle - first +1
for(k = 0 and l equal to middle + 1 to last, incrementing both by 1) do the following
thisRight of position K equals new country
Populate thisRightposition K with thisList position l
End for
Call mergeSortCountries for thisLeft using thisLeft, 0, and Middle
Call mergeLeftRight for thisList using thisList, thisLeft, and thisRight
STOP

B6. The mergeLeftRight(Country[] thisList, Country[] leftList, Country[] rightList) Method
START
Declare x, y,z,first as integers
Set first = 0
Declare last as an integer equal to leftList.length + rightList.length -1
for(x equal to y equal to z equal to first, until z equals last, increment x y and z)
Do the following
If x is greater than or equal to leftList.length thisList position z equals rightList position y
Else if y >= rightList.length then thisList position z equals leftList position x
else
If leftList position x country number is less than that of rightList position y then
thisList position z equals leftList position x
Decrement y
Else thisList position z equals rightList position y
Decrement x
End if
End for

Return thisList
STOP

B7. The bubbleSort(Country[] thisList) Method
START

Try
Call loadCountries for thisList
Catch
Exception ex
Display error message
Declare dummyCountry equal to a new country
Declare x, and pass as integers
Declare int thisLimit to the length of thisList
Declare isSorted as a boolean
Declare startTime, endTime and compTime as doubles
Call system.nanoTime for startTime
for(pass equal to 1 to thisLimit incrementing by 1) do the following
Set isSorted to true
for (x equal to 1 to thisLimit - pass incrementing by 1) do the following
If thisList position x-1 country code is greater than thisList position x country code
then
Set dummyCountry equal to thisList position x-1
Set thisList x-1 to thisList position x
Set thisList position x to dummyCountry
Set isSorted to false
End if
End for
If isSorted break
End for
Call system.nanoTime for endTime
Set compTime to the difference between endTime and startTime
Call listCountries with thisList and compTime
STOP

B8. The exchangeSelection(Country[] thisList) Method
START

Try
Call loadCountries for thisList
Catch
Exception ex
Display error message
Declare dummyCountry equal to a new country
Declare counter 1 and counter 2 to integers
Declare thisLimit to an integer equal to the length of thisList
Declare startTime, endTime and compTime as doubles
Call system.nanoTime for startTime

for(counter 1 equal to 1 to thisLimit incrementing by 1) do the following
for(counter 2 equal to counter1 + 1 to thisLimit incrementing by 1)
Do the following
If thisList position counter1-1 > thisList position counter2-1 then
Set dummyCountry to thisList at position counter1-1
Set thisList position counter1-1 to thisList position counter2-1
Set thisList position counter2-1 to dummyCountry
End if
End for
End for
Call system.nanoTime for endTime
Set compTime to the difference between endTime and startTime
Call listCountries with thisList and compTime

STOP

B9. The straightSelection(country[] thisList) Method

START

Try
Call loadCountries for thisList
Catch
Exception ex
Display error message
Declare dummyList equal to a new country country array of length thisList
Declare x, and y to integers
Declare index to an integer equal to 0
Declare smallest to an integer equal to MAX_VALUE
Declare thisLimit to an integer equal to the length of thisList
Declare startTime, endTime and compTime as doubles
Call system.nanoTime for startTime
for(y = 0 to thisList length incrementing by 1) do the following
for(x = 0 to thisList length incrementing by 1) do the following
If thisList position x-1 country code is less than smallest then
Set smallest to country code of thisList position x-1
Set index to x-1
Call destroyMe for object at thiList position x-1
End if
End for
Set dummyList position y-1 to thisList entry at position index
End for
Call system.nanoTime for endTime
Set compTime to the difference between endTime and startTime
Call listCountries with thisList and compTime

STOP

B10. The quickSortController(Country[] thisList) Method

START

Declare startTime, endTime and compTime as doubles
Call system.nanoTime for startTime
Try
Set thisList equal to loadCountries
Catch
Exception ex
Call quickSortCountries for thisList using thisList, 0, and thisList.length-1
Call system.nanoTime for endTime
Set compTime to the difference between endTime and startTime
Call listCountries with thisList and compTime
STOP

B11. The quickSortCountries(Country[] thisList, int first, int last) method
START

Declare dummyCountry to a country
Declare k,l, and p as integers
Declare boolean isSorted as false
If length of thisList is 1 then isSorted equals true
If length of thisList is 2 and the country code at position 0 is less than the country code of thisList at position 1
Then isSorted equals true
If isSorted is false then
Set k equal to first
Set l equal to last
Set p equal to (first + last) /2
Set dummyCountry equal to a new Country
While k is less than or equal to l
While the country code of thisList a position k is less than the country code of thisList at position p
Increment k
End while
While the country code of thisList a position l is greater than the country code of thisList at position p
Decrement l
End while
If k is less than or equal to l then
Call modifyMe for dummyCountry with thisList at position k
Call modifyMe for thisList position l with thisList at position l
Call modifyMe for thisList position l with dummyCountry
End if
End while
If first is less than l then
Call quickSortCountries for thisList using thisList, first, and p
End if
If k is less than last then

Call quickSortCountries for thisList using thisList, p, and last
End if
Return thisList
STOP

B12. The vToArray(Vector <Country> thisVector) Method
START
Declare x, and numCountries as integers
Declare thisList as a new country array equal to null
Set numCountries equal to thisVector.size
Set thisList equal to a new country array of size numCountries
for(x = 1 to numCountries incrementing by 1) do the following
thisList position x -1 equals new country
thisList position x-1 equals thisVector position x-1
End for
Return thisList
STOP

B13. The loadCountries() throws Exception, IOException Method
START
Declare thisList as a country array
Try
Set CountryFileReader to a new FileInputStream using COuntryFileString
Set CountryObjectReader to new ObjectInputStream using CountryFileReader
Set thisList to CountryObjectReader.readObject cast to COuntry array
Catch
IOException EX with error message dialog
Throw EX
IO Exception EX2 with error message dialog
Throw EX2
Return thisList
STOP

B14. The createFile() Method
START
Set createFile to new File using CountryFileString
If createFile.createNewFile then
Show Message stating file has been created
Else
Show message stating file already exists
End if
Catch
IOException EX1
Show message dialog stating error
STOP

B15. The deleteFile() Method
START

Try
Declare myObj to new File using CountryFileString
If myObj.delete then
Show message stating file has been deleted
Else
Show message stating the file failed to delete
End if
Catch
IOException EX1
Show message dialog stating error
STOP