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 CountriesMonitor Class

CountriesMonitor

public static Stack countriesList;
public static final String HEADING = "Countries Stack of Petr Bowles";
public static final int DEFAULT_NUMBER = 0;
static double totalPCI, averagePCI, stdDevPCI;

public static void main(String[] args) public static void inputCountries() public static void queryCountry(Stack thisList) public static void listCountries(Stack thisList) public static void sortCountries (Stack thisList) public static void standardDev (Stack ThisList, int thisLim, double thisAvg); public static void removeCountries () public static void checkSize(Stack thisList) public static void initialize() public static void empty()

```
Public static void summarizeCountries(Stack thisList);
Private static void highLow(Country thisCountry();
Public static void initializeSummary();
```

```
        B0. The main(String[] argos) Method
START
        Let exitTime be Boolean, initialized to FALSE;
        Let option be an integer;
        Let countryList be an Stack of Country objects;
        totalPCI = 0;

        // Main Operations
        While (Not exitTime) do the following:
        Present the user with the following menu:
        1. Enter Countries Info
        2. Query for a Country
        3. List unsorted countries
        4. Remove a country
        5. Check the size of the list
        6. Empty the list
        7. Prove the summary
        8. Display the sorted list
        9. Exit Prompt user to key in the a menu selection and store this in Option;

        Case Option is:
        1: countryList := inputCountries ( ); // Obtain information for countries
        2: queryCountry(countriesList);
        3: listCountries(countriesList);
        4: removeCountries(countriesList);
        5: checkSize(countriesList);
        6. empty();
        7. standardDev(countriesList, averagePCI);
        8. sortCountries(countriesList);
        9: Set exitTime to True;

        End-Case;
        End-While;
STOP
        B1.The inputCountries() Method
START
        Let x, numberOfCountries be integers;
        Let currentCountry be a new country object;

        Prompt for number of countries and store this in numberOfCountries;
        Ensure capacity of countriesList by comparing to numberOfCountries;
        For (x := 1 to numberOfCountries with increments of 1) do the following
        Instantiate currentCountry = new Country()];
        currentCountry.inputData(x);
        countriesList.push(x-1, currentCountry);
        totalPCI += currentCountry.getPCI();

        End-For;

        averagePCI := totalPCI/numberOfCountries;
STOP
        B2. The queryCountry(Stack <Country> thisList) Method
START
        Let outString be a string;
        Let searchCountry and foundCountry be new Country Objects

        String qHeading = "Country Query";
        boolean exitTime = false;
        boolean exitNow;
        int thisLim = thisList.getSize();

        If thisLim > 0:
        While exitTime is true;
        Accept user input for searchNumber;
        Declare foundCountry as a new empty Country;
        exitNow = false;

        For(x := 1 to thisLim && exitNow is false,  incrementing by 1) do the following:
        If searchNumber match thisList.getInfo(x-1.getCountry());
        Set foundCountry equal to the value and exitTime to true;
        End if;

        Otherwise set outString to error message
        Prompt for nextUserAction
        If nextUserAction equals cancel option, then exitTime = true
        End while
        End if
STOP
        B3. The listCountries(CountryStack thisList) Method
START
        Let outString be a string = "Members of the list are:"
        for(x:= 1 to thisList.size() incrementing by 1) do the following:
        Add thisList.get(x-1).printMe() to outString
        End for
        Show output
STOP

        B4. The sortCountries(Stack <Country> thisList) Method
START
        Let outString be a string = “Members of the list are:”
        Let sortedCountry be a new Country;
        Let newList be and ArrayList equal to thisList;
        Int limit = newList.size();

        for(x:= 0 to limit - 1 incrementing by 1) do the following:

        for(y:= 0 to limit - 1 incrementing by 1) do the following:
        If newList position x PCI is greater than newList position y + 1 PCI do the following:
        sortedCountry.modifyMe(newList.get(y + 1));//x
        newList.get(y + 1).modifyMe(newList.get(x));//x,y
        newList.get(x).modifyMe(sortedCountry);//x,sort

        End if
        End for
        End for

        for(z := 1 to limit incrementing by 1) do the following:
        outString += newList.getInfo(z-1).printMe();
        End for
        Output outString;
STOP
        B5. The standardDev(Stack <Country> thisList, int thisLim, double thisAvg) Method
START
        double standard, diff, totalDiff,
       int limit = thisList.size();
       totalDiff = 0;

        for(x := 1 to limit incrementing by 1) do the following:
        diff = (thisList.get(x-1).getPCI() - averagePCI)^2;
        totalDiff += diff;
        End for

       standard = Math.sqrt(totalDiff/limit);
        Return standard
STOP
        B6. The removeCountries() Method
START
        Let removalPrompt and removalHeading be strings;
        removalHeading = "Removal of Items from the List";
        Let x, rStart, rStop, popAmount, and nextUserAction be integers;

        Accept user input for rStart, popAmount and rStop;

        while(popAmount > countriesList.getSize() or popAmount <= 0) do the following:
        Show error message;
        Accept input for rStart and rStop;
        End while

        removalPrompt = popAmount + rStart + " to " + rStop + " are about to be removed from
the list.\n" + "Click Yes to remove the items. Click No or Cancel to exit.";
        Accept user input for nextUserAction;

        if(nextUserAction = Yes Option) do the following:
        for(x = popAmount incrementing by 1) do the following:
        countriesList.pop();

        End for
        End if

STOP
        B7. The checkSize(Stack <Country> thisList) Method
START
        Let Output be a JOptionPane output message;
        Output = "There are " + thisList.getSize() + " countries in the list";
STOP
        B8. The initializeList() Method
START
        countriesList = new CountryStack ();
STOP
        B9. The empty() Method
START
        Let x and nextUserAction be integers;
        Let removalPrompt be a string;
        removalPrompt = "You are about to empty the list. " + "Click Yes to Empty. Click No or
        Cancel to exit.";
        nextUserAction = JOptionPane.showConfirmDialog(null, removalPrompt);
        If nextUserAction == JOptionPane.YES_OPTION do the following:
        countriesList.clearStack();
        End if;
STOP
        B10. The SummarizeCountries(Stack <Country> thisList) Method
START
```

```
        Let x cLim be integers;
        Let thisList be a Stack of country objects
        Let outputS be a string initialized to blanks
        Set cLim to the size of thisList
        initializeSummary();
        for(x = 1 to cLim with increments of 1) do the following
        Add thisList.get(x -1).getPCI() to totalPCI
        highLow(thisList.getInfo(x-1)
        End for
        averagePCI = totalPCI/cLim
        stdDevPCI = standardDev(thisList, cLim, averagePCI)
        Append richestC.printMe() to outputS
        Append poorestC.printMe() to outputS
        Append “total PCI: “ + totalPCI to outputS
        Append “average PCI: “ + averagePCI to outputS
        Append “standard deviation: “ + stdDevPCI to outputS
        Display(outputS)
STOP
        B11. The highLow(Country thisCountry) Method
START
        Let thisCountry be a Country instance;
        If thisCountry.getPCI() > richestC.getPCI()
        richestC.modifyMe(thisCountry)
        End if
        If thisCountry.getPCI() < poorestC.getPCI()
        poorestC.modifyMe(thisCountry)
        End if
STOP
        B12. The initializeSummary() Method
START
        Instantiate richestC, poorestC
        richestC.cPCI to smallest possible value
        poorestC.cPCI to largest possible value
        Set totalPCI, averagePCI, stdDevPCI to zero
STOP
```