Exercise 1

```
/**
	Saket Bakshi. 4/23/19. Period 6. This is used for problem 1 of Chapter 14.
	Creates random arrays to sort and manipulates arrays.
*/
import java.util.Random;
public class ArrayUtil
{
	public static Random generator = new Random();

	/**
		Creates an array filled with random values.
		@param length the length of the array
		@param n the number of possible random values
		@return an array filled with numbers between 0 and n-1
	*/
	public static int[] randomIntArray(int length, int n)
	{
		int[] a = new int[length];
		for(int i = 0; i < a.length; i++)
		{
			a[i] = generator.nextInt(n);
		}
		return a;
	}

	/**
		Swaps two entries of an array.
		@param a the array
		@param i the first position to swap
		@param j the second position to swap
	*/
	public static void swap(int[] a, int i, int j)
	{
		int temp = a[i];
		a[i] = a[j];
		a[j] = temp;
	}
}

/**
	Saket Bakshi. 4/23/19. Period 6. This is used for problem 1 of Chapter 14.
```

```
	Modifies selection sort to sort integers in descending order.
*/
public class SelectionSortReverse
{
	/**
		Sorts an array, using selection sort.
		@param a the array to sort
	*/
	public static void sort(int[] a)
	{
		for(int i = 0; i < a.length - 1; i++)
		{
			int maxPos = maximumPosition(a, i);
			ArrayUtil.swap(a, maxPos, i);
		}
	}

	/**
		Finds the largest element in a tail range of the array.
		@param a the array to sort
		@param from the first position in a to compare
		@return the position of the largest element in the range a[from] . . . a[a.length-1]
	*/
	private static int maximumPosition(int[] a, int from)
	{
		int maxPos = from;
		for(int i = from + 1; i < a.length; i++)
		{
			if(a[i] > a[maxPos]) {maxPos = i;}
		}
		return maxPos;
	}
}

/**
	Saket Bakshi. 4/23/19. Period 6. This is used for problem 1 of Chapter 14.
	Modifies selection sort to sort integers in descending order.
*/
import java.util.Arrays;
public class SelectionReverseDemo
{
	public static void main(String[] args)
	{
```

```
        int[] a = ArrayUtil.randomIntArray(20,100);
        System.out.println(Arrays.toString(a));

        SelectionSortReverse.sort(a);

        System.out.println(Arrays.toString(a));
    }
}
```

```
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C14EXBakshiSaket\PracticeExercisesCh14E1> java SelectionReverseDemo
[41, 3, 63, 8, 38, 68, 42, 18, 88, 26, 57, 84, 3, 18, 20, 49, 13, 35, 39, 79]
[88, 84, 79, 68, 63, 57, 49, 42, 41, 39, 38, 35, 26, 20, 18, 18, 13, 8, 3, 3]
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C14EXBakshiSaket\PracticeExercisesCh14E1>
```

Exercise 2

```
/**
	Saket Bakshi. 4/23/19. Period 6. This is used for problem 2 of Chapter 14.
	Creates coin objects.
*/
public class Coin
{
	private double value;
	private String name;

	/**
		Creates a coin object.
	*/
	public Coin()
	{
		value = 0;
		name = "";
	}

	/**
		Creates a coin object.
		@param a the value
		@param b the name
	*/
	public Coin(double a, String b)
	{
		value = a;
		name = b;
	}

	/**
		Gets the value of the coin.
		@return the value
	*/
	public double getValue()
	{
		return value;
	}

	/**
		Gets the name of the coin.
		@return the name
```

```
	*/
	public String getName()
	{
		return name;
	}
}

/**
	Saket Bakshi. 4/23/19. Period 6. This is used for problem 2 of Chapter 14.
	Manipulates coin arrays.
*/
public class ArrayUtilCoin
{
	/**
		Swaps two entries of an array.
		@param a the array
		@param i the first position to swap
		@param j the second position to swap
	*/
	public static void swap(Coin[] a, int i, int j)
	{
		Coin temp = a[i];
		a[i] = a[j];
		a[j] = temp;
	}
}

/**
	Saket Bakshi. 4/23/19. Period 6. This is used for problem 2 of Chapter 14.
	Modifies selection sort to sort coins by their value.
*/
public class SelectionSortCoins
{
	/**
		Sorts an array, using selection sort.
		@param a the array to sort
	*/
	public static void sort(Coin[] a)
	{
		for(int i = 0; i < a.length - 1; i++)
		{
			int minPos = minimumPosition(a, i);
			ArrayUtilCoin.swap(a, minPos, i);
```

```
                }
        }

        /**
                Finds the smallest value of a coin in a tail range of the array.
                @param a the array to sort
                @param from the first position in a to compare
                @return the position of the smallest coin value in the given range
        */
        private static int minimumPosition(Coin[] a, int from)
        {
                int minPos = from;
                for(int i = from + 1; i < a.length; i++)
                {
                        if(a[i].getValue() < a[minPos].getValue()) {minPos = i;}
                }
                return minPos;
        }
}

/**
        Saket Bakshi. 4/23/19. Period 6. This is used for problem 2 of Chapter 14.
        Tests a selection sort of coins.
*/
import java.util.Arrays;
public class SelectionCoinDemo
{
        public static void main(String[] args)
        {
                Coin penny = new Coin(0.01, "penny");
                Coin nickel = new Coin(0.05, "nickel");
                Coin dime = new Coin(0.1, "dime");
                Coin quarter = new Coin(0.25, "quarter");

                Coin[] a = {penny, quarter, quarter, penny, dime, nickel, penny, penny, quarter,
dime};

                System.out.print("Coin values: ");
                for(int i = 0; i < a.length - 1; i++)
                {
                        System.out.print(a[i].getName() + ", ");
                }
                System.out.print(a[a.length-1].getName());
```

```
            SelectionSortCoins.sort(a);

            System.out.println();

            System.out.print("Coin values: ");
            for(int i = 0; i < a.length - 1; i++)
            {
                  System.out.print(a[i].getName() + ", ");
            }
            System.out.print(a[a.length-1].getName());
      }
}
```

```
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C14EXBakshiSaket\PracticeExercisesCh14E2> java SelectionCoinDemo
Coin values: penny, quarter, quarter, penny, dime, nickel, penny, penny, quarter, dime
Coin values: penny, penny, penny, penny, nickel, dime, dime, quarter, quarter, quarter
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C14EXBakshiSaket\PracticeExercisesCh14E2>
```

Exercise 4

```
/**
	Saket Bakshi. 4/23/19. Period 6. This is used for problem 1 of Chapter 14.
	Modifies merge sort to sort Strings.
*/
public class MergeSorterString
{
	/**
		Sorts a String array lexographically, using merge sort.
		@param a the array to sort
	*/
	public static void sort(String[] a)
	{
		if(a.length <= 1) {return;}
		String[] first = new String[a.length/2];
		String[] second = new String[a.length - first.length];
		for(int i = 0; i < first.length; i++)
		{
			first[i] = a[i];
		}
		for(int i = 0; i < second.length; i++)
		{
			second[i] = a[first.length + i];
		}
		sort(first);
		sort(second);
		merge(first, second, a);
	}

	/**
		Merges two sorted arrays into an array.
		@param first the first sorted array
		@param second the second sorted array
		@param a the array into which to merge first and second
	*/
	public static void merge(String[] first, String[] second, String[] a)
	{
		int iFirst = 0;
		int iSecond = 0;
		int j = 0;

		while(iFirst < first.length && iSecond < second.length)
```

```
		{
			if(first[iFirst].compareTo(second[iSecond]) < 0)
			{
				a[j] = first[iFirst];
				iFirst++;
			}
			else
			{
				a[j] = second[iSecond];
				iSecond++;
			}
			j++;
		}

		while(iFirst < first.length)
		{
			a[j] = first[iFirst];
			iFirst++;
			j++;
		}

		while(iSecond < second.length)
		{
			a[j] = second[iSecond];
			iSecond++;
			j++;
		}
	}
}

/**
	Saket Bakshi. 4/23/19. Period 6. This is used for problem 4 of Chapter 14.
	Tests String Merge sort.
*/
import java.util.Scanner;

public class MergeStringDemo
{
	public static void main(String[] args) {
		String[] names = {"John", "Nathan", "Saket", "Nic", "Arthur", "Ian", "Aris", "Chris",
"Takeru", "Max", "Thinh", "Hung", "Daniel", "Josie", "Randy"};

		System.out.print("Original array: ");
```

```
		for(int i = 0; i < names.length - 1; i++)
		{
			System.out.print(names[i] + ", ");
		}
		System.out.print(names[names.length-1]);

		MergeSorterString.sort(names);

		System.out.print("\nSorted array: ");
		for(int i = 0; i < names.length - 1; i++)
		{
			System.out.print(names[i] + ", ");
		}
		System.out.print(names[names.length-1]);
	}
}
```

```
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C14EXBakshiSaket\PracticeExercisesCh14E4> java MergeStringDemo
Original array: John, Nathan, Saket, Nic, Arthur, Ian, Aris, Chris, Takeru, Max, Thinh, Hung, Daniel, Josie, Randy
Sorted array: Aris, Arthur, Chris, Daniel, Hung, Ian, John, Josie, Max, Nathan, Nic, Randy, Saket, Takeru, Thinh
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C14EXBakshiSaket\PracticeExercisesCh14E4>
```

Project 3

```
/**
   Saket Bakshi. 4/23/19. Period 6. This is used for project 3 of Chapter 14.
   Does radix style arrays for integers up to 999
*/
import java.util.Arrays;
import java.util.Scanner;

public class Radix
{
   public static void main(String[] args)
   {
      Scanner in = new Scanner(System.in);
      System.out.print("Enter array size: ");
      int n = in.nextInt();

      int[] a = ArrayUtil.randomIntArray(n, 999);
      n = a.length;

      System.out.print("Original array: ");
      for(int i = 0; i < a.length - 1; i++)
      {
         System.out.print(a[i] + ", ");
      }
      System.out.print(a[a.length-1]);

      StopWatch timer = new StopWatch();

      timer.start();
      radixsort(a, n);
      timer.stop();


      System.out.print("\nNew array: ");
      for(int i = 0; i < a.length - 1; i++)
      {
         System.out.print(a[i] + ", ");
      }
      System.out.print(a[a.length-1]);
      System.out.println("\nElapsed time: " + timer.getElapsedTime() + " milliseconds");

   }
```

```
    public static void countSort(int arr[], int n, int exp)
    {
        int[] output = new int[n];
        int[] count = new int[10];
        int i;
        Arrays.fill(count, 0);

        for(i = 0; i < n; i++) {count[(arr[i] / exp) % 10]++;}

        for(i = 1; i < 10; i++) {count[i] += count[i - 1];}

        for(i = n - 1; i >= 0; i--)
        {
            output[count[(arr[i] / exp) % 10] - 1] = arr[i];
            count[(arr[i] / exp) % 10]--;
        }

        for(i = 0; i < n; i++) {arr[i] = output[i];}
    }

        public static int getMax(int arr[], int n)
    {
        int max = arr[0];
        for(int i = 1; i < n; i++)
            if(arr[i] > max)
                max = arr[i];
        return max;
    }

    public static void radixsort(int arr[], int n)
    {
        int m = getMax(arr, n);
        for(int exp = 1; m/exp > 0; exp *= 10) {countSort(arr, n, exp);}
    }
}

/**
        Saket Bakshi. 4/23/19. Period 6. This is used for project 3 of Chapter 14.
        Creates random arrays. Manipulates arrays.
*/
import java.util.Random;
public class ArrayUtil
```

```
{
	public static Random generator = new Random();

	/**
		Creates an array filled with random values.
		@param length the length of the array
		@param n the number of possible random values
		@return an array filled with numbers between 0 and n-1
	*/
	public static int[] randomIntArray(int length, int n)
	{
		int[] a = new int[length];
		for(int i = 0; i < a.length; i++)
		{
			a[i] = generator.nextInt(n);
		}
		return a;
	}

	/**
		Swaps two entries of an array.
		@param a the array
		@param i the first position to swap
		@param j the second position to swap
	*/
	public static void swap(int[] a, int i, int j)
	{
		int temp = a[i];
		a[i] = a[j];
		a[j] = temp;
	}
}

public class StopWatch
{
	private long elapsedTime;
	private long startTime;
	private boolean isRunning;

	public StopWatch()
	{
		reset();
	}
```

```
	public void start()
	{
		if(isRunning) {return;}
		isRunning = true;
		startTime = System.currentTimeMillis();
	}

	public void stop()
	{
		if(!isRunning) {return;}
		isRunning = false;
		long endTime = System.currentTimeMillis();
		elapsedTime = elapsedTime + endTime - startTime;
	}

	public long getElapsedTime()
	{
		if(isRunning)
		{
			long endTime = System.currentTimeMillis();
			return elapsedTime + endTime - startTime;
		}
		else
		{
			return elapsedTime;
		}
	}

	public void reset()
	{
		elapsedTime = 0;
		isRunning = false;
	}
}
```

```
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C14EXBakshiSaket\PracticeExercisesCh14P3> java Radix
Enter array size: 150
Original array: 307, 719, 398, 54, 328, 549, 273, 440, 935, 913, 371, 870, 118, 650, 150, 6, 624, 447, 260, 371, 287, 381, 728, 746, 9
44, 51, 846, 141, 187, 329, 737, 682, 661, 199, 160, 798, 748, 82, 875, 99, 270, 49, 416, 609, 454, 526, 884, 729, 37, 79, 328, 276, 6
57, 20, 818, 218, 12, 676, 261, 734, 44, 896, 238, 582, 322, 937, 595, 673, 945, 805, 890, 945, 221, 116, 868, 704, 662, 312, 437, 335
, 680, 499, 592, 474, 887, 108, 507, 343, 180, 721, 44, 605, 719, 368, 359, 713, 775, 408, 69, 619, 734, 34, 514, 639, 122, 717, 515,
152, 473, 623, 130, 863, 876, 823, 464, 393, 910, 479, 30, 710, 554, 544, 360, 405, 863, 309, 471, 30, 201, 569, 841, 736, 404, 341, 8
54, 616, 447, 930, 650, 179, 963, 772, 754, 274, 503, 63, 789, 53, 986, 875
New array: 6, 12, 20, 30, 30, 34, 37, 44, 44, 49, 51, 53, 54, 63, 69, 79, 82, 99, 108, 116, 118, 122, 130, 141, 150, 152, 160, 179, 18
0, 187, 199, 201, 218, 221, 238, 260, 261, 270, 273, 274, 276, 287, 307, 309, 312, 322, 328, 328, 329, 335, 341, 343, 359, 360, 368, 3
71, 371, 381, 393, 398, 404, 405, 408, 416, 437, 440, 447, 447, 454, 464, 471, 473, 474, 479, 499, 503, 507, 514, 515, 526, 544, 549,
554, 569, 582, 592, 595, 605, 609, 616, 619, 623, 624, 639, 650, 650, 657, 661, 662, 673, 676, 680, 682, 704, 710, 713, 717, 719, 719,
 721, 728, 729, 734, 734, 736, 737, 746, 748, 754, 772, 775, 789, 798, 805, 818, 823, 841, 846, 854, 863, 863, 868, 870, 875, 875, 876
, 884, 887, 890, 896, 910, 913, 930, 935, 937, 944, 945, 945, 963, 986
Elapsed time: 0 milliseconds
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C14EXBakshiSaket\PracticeExercisesCh14P3> java Radix
Enter array size: 50
Original array: 856, 306, 912, 100, 693, 710, 163, 903, 100, 186, 30, 988, 189, 167, 787, 108, 46, 619, 138, 544, 931, 824, 731, 526,
738, 194, 253, 959, 185, 485, 496, 854, 942, 860, 444, 571, 691, 22, 852, 469, 397, 347, 450, 543, 319, 253, 687, 473, 48, 762
New array: 22, 30, 46, 48, 100, 100, 108, 138, 163, 167, 185, 186, 189, 194, 253, 253, 306, 319, 347, 397, 444, 450, 469, 473, 485, 49
6, 526, 543, 544, 571, 619, 687, 691, 693, 710, 731, 738, 762, 787, 824, 852, 854, 856, 860, 903, 912, 931, 942, 959, 988
Elapsed time: 0 milliseconds
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C14EXBakshiSaket\PracticeExercisesCh14P3> java Radix
Enter array size: 15
Original array: 517, 609, 94, 846, 114, 1, 343, 307, 772, 37, 702, 646, 22, 780, 797
New array: 1, 22, 37, 94, 114, 307, 343, 517, 609, 646, 702, 772, 780, 797, 846
Elapsed time: 0 milliseconds
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C14EXBakshiSaket\PracticeExercisesCh14P3>
```