Exercise 1

```
/**
	Saket Bakshi. 4/7/19. Period 6. This is used for problem 1 of Chapter 13.
	Creates a rectangle class with a recursive getArea method.
*/
public class RectangleV1
{
	private int width;
	private int height;

	/**
		Creates a rectangle object.
		@param width the width of the rectangle
		@param height the height of the rectangle
	*/
	public RectangleV1(int width, int height)
	{
	   this.width = width;
	   this.height = height;
	}

	/**
		Calculates the area of the rectangle recursively.
		@return the area
	*/
	public int getArea()
	{
		if(this.width <= 0 || this.height <=0)
		{
			return 0;
		}
		else if(width == 1)
		{
			return height;
		}
		else
		{
		RectangleV1 smallerRectangle = new RectangleV1(width-1, height);
		int smallerArea = smallerRectangle.getArea();
		return smallerArea + height;
		}
	}
```

```
}

/**
	Saket Bakshi. 4/7/19. Period 6. This is used for problem 1 of Chapter 13.
	Tests a rectangle class with a recursive getArea method.
*/
public class RectangleV1Tester
{
	public static void main(String[] args)
	{
		final int WIDTH = 20; //height and widths of rectangles
		final int HEIGHT= 8;

		RectangleV1 rectangle1 = new RectangleV1(WIDTH, HEIGHT); //creating
objects
		RectangleV1 rectangle2 = new RectangleV1(WIDTH-1, HEIGHT);
		System.out.println("Area, first rectangle: " + rectangle1.getArea()); //experimental
calculations
		System.out.println("Expected: 160"); //expected
		System.out.println("Area of second rectangle: " + rectangle2.getArea());
		System.out.println("Expected: 152");
	}
}
```

```
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C13EXBakshiSaket\PracticeExercisesCh13E1> java RectangleV1Tester
Area, first rectangle: 160
Expected: 160
Area of second rectangle: 152
Expected: 152
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C13EXBakshiSaket\PracticeExercisesCh13E1>
```

Exercise 2

```
/**
	Saket Bakshi. 4/7/19. Period 6. This is used for problem 2 of Chapter 13.
	Creates a square class with a recursive getArea method.
*/
public class SquareV1
{
	private int width;

	/**
		Creates a square object.
		@param width the width of the square
	*/
	public SquareV1(int width)
	{
	this.width = width;
	}

	/**
		Calculates the area of the square recursively.
		@return the area
	*/
	public int getArea()
	{
	if(width <= 0)
	{
		return 0;
	}
	else if(width == 1)
	{
		return 1;
	}
	else
	{
		SquareV1 smallerSquare = new SquareV1(width-1);
		int smallerArea = smallerSquare.getArea();
		return smallerArea + width + width-1;
	}
	}
}

/**
```

```
	Saket Bakshi. 4/7/19. Period 6. This is used for problem 2 of Chapter 13.
	Tests a square class with a recursive getArea method.
*/
public class SquareV1Tester
{
	public static void main(String[] args)
	{
		final int WIDTH = 20; //width of the first square

		SquareV1 square1 = new SquareV1(WIDTH); //creating objects
		SquareV1 square2 = new SquareV1(WIDTH-1);
		System.out.println("Area, first square: " + square1.getArea()); //experimental
calculations
		System.out.println("Expected: 400"); //expected
		System.out.println("Area, second square: " + square2.getArea());
		System.out.println("Expected: 361");
	}
}
```

```
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C13EXBakshiSaket\PracticeExercisesCh13E2> java SquareV1Tester
Area, first square: 400
Expected: 400
Area, second square: 361
Expected: 361
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C13EXBakshiSaket\PracticeExercisesCh13E2>
```

Exercise 3

```
/**
	Saket Bakshi. 4/7/19. Period 6. This is used for problem 3 of Chapter 13.
	Reverses a string recursively.
*/
public class StringReverse
{
	public static void main(String[] args)
	{
		System.out.println("First string: \"Hello!\"");
		System.out.println("\nResults: \"" + reverse("Hello!") + "\"");
		System.out.println("Expected: \"!olleH\"");
	}

	/**
		Reverses the order of a String, recursively.
		@param text the String to reverse
		@return the reversed String
	*/
	public static String reverse(String text)
	{
		if(text.length() == 1)
			return text;
		else
			return reverse(text.substring(1)) + text.substring(0,1);
	}
}
```

```
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C13EXBakshiSaket\PracticeExercisesCh13E3> java StringReverse
First string: "Hello!"

Results: "!olleH"
Expected: "!olleH"
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C13EXBakshiSaket\PracticeExercisesCh13E3>
```

Project 2

```
/**
        Saket Bakshi. 4/7/19. Period 6. This is used for project 2 of Chapter 13.
        This class computes the permutations of an integer array. This is used to permute Strings, which have problems with repeated characters. By basing the Strings off of their character position on an integer array, we can truly permute any String.
*/
public class PermutationCalculator
{
        private int[] arr;
   private boolean done;

        /**
      Creates an array object to be permuted.
      @param n the length of the array to be permutted
   */
   public PermutationCalculator(int n)
   {
      arr = new int[n];
      for (int i = 0; i < n; i++)
      {
         arr[i] = i;
      }
   }

   /**
      Sends the next permutation of number orders.
      @return the next permutation
   */
   public int[] nextPer()
   {
      if (arr.length <= 1)
      {
         return arr;
      }
      for (int i = arr.length - 1; i > 0 ; i--)
      {
         if (arr[i - 1] < arr[i])
         {
            int j = arr.length - 1;
            while (arr[i - 1] > arr[j])
            {
```

```
            j--;
         }
         swap(i - 1, j);
         reverse(i, arr.length - 1);
         return arr;
      }
   }
   return arr;
}

/**
   Checks if there are more permutations to return.
   @return if there are more permutations
*/
public boolean canPermuteMore()
{
   if (arr.length <= 1)
   {
      return false;
   }
   for (int i = arr.length - 1; i > 0; i--)
   {
      if (arr[i - 1] < arr[i])
      {
         return true;
      }
   }
   return false;
}

/**
   Swaps two items in an array.
   @param i the position of the first element to be swapped
   @param j the position of the second element to be swapped
*/
public void swap(int i, int j)
{
   int temp = arr[i];
   arr[i] = arr[j];
   arr[j] = temp;
}

/**
```

```
      Reverses the order of an array from position i to position j
      @param i the first position
      @param j the second position
   */
   public void reverse(int i, int j)
   {
      while (i < j)
      {
         swap(i, j);
         i++;
         j--;
      }
   }
}

/**
      Saket Bakshi. 4/7/19. Period 6. This is used for project 2 of Chapter 13.
      This class permutes Strings by calculating permutations of their character positions.
*/
public class PermutationOrganizer
{
      private String word;
   private PermutationCalculator example;

   /**
      Organizes a PermutationCalculator object to return permutations of a String.
      @param aWord the word to permute
   */
   public PermutationOrganizer(String aWord)
   {
      word = aWord;
      example = new PermutationCalculator(word.length());
   }

   /**
            Sees if more permutations can be calculated.
            @return if more permutations can be done
      */
   public boolean canPermuteMore()
   {
      return example.canPermuteMore();
   }
```

```
   /**
         Calculates the next permutation.
         @return the next permutation
   */
   public String nextPermutation()
   {
      int[] arr = example.nextPer();

      String e = "";
      for (int i = 0; i < arr.length; i++)
      {
         e = e + word.charAt(arr[i]);
      }
      return e;
   }
}

/**
         Saket Bakshi. 4/7/19. Period 6. This is used for project 2 of Chapter 13.
         This class tests the calculation of permutations of a String.
*/
public class PermutationTester
{
         public static void main(String[] args)
         {
                  PermutationOrganizer test = new PermutationOrganizer("bruh");
                  System.out.println("bruh");

      while (test.canPermuteMore())
      {
         System.out.println(test.nextPermutation());
      }
         }
}
```

```
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C13EXBakshiSaket\PracticeExercisesCh13P2> java PermutationTester
bruh
brhu
burh
buhr
bhru
bhur
rbuh
rbhu
rubh
ruhb
rhbu
rhub
ubrh
ubhr
urbh
urhb
uhbr
uhrb
hbru
hbur
hrbu
hrub
hubr
hurb
PS C:\Users\saket\Git\CSWork\JAVA\ChapterAssignments\C13EXBakshiSaket\PracticeExercisesCh13P2>
```