Assignment 2 – Kyle Guarco

Output

```
*** *** *** *** *** *** *** *** *** ***          (idxred = -1, idxblue = 10, stacksize = 0)
1** *** *** *** *** *** *** *** *** 10*          (idxred = 0, idxblue = 9, stacksize = 2)
1** 2** *** *** *** *** *** *** 20* 10*          (idxred = 1, idxblue = 8, stacksize = 4)
Popping from the red stack: 2
1** *** *** *** *** *** *** *** 20* 10*          (idxred = 0, idxblue = 8, stacksize = 3)
Popping from the blue stack: 20
1** *** *** *** *** *** *** *** *** 10*          (idxred = 0, idxblue = 9, stacksize = 2)
Peeking at the blue stack: 30
1** 3** *** *** *** *** *** *** 30* 10*          (idxred = 1, idxblue = 8, stacksize = 4)
1** 3** *** *** *** *** *** 40* 30* 10*          (idxred = 1, idxblue = 7, stacksize = 5)
1** 3** 4** *** *** *** *** 40* 30* 10*          (idxred = 2, idxblue = 7, stacksize = 6)
1** 3** 4** *** *** *** 50* 40* 30* 10*          (idxred = 2, idxblue = 6, stacksize = 7)
1** 3** 4** 5** *** *** 50* 40* 30* 10*          (idxred = 3, idxblue = 6, stacksize = 8)
1** 3** 4** 5** *** 60* 50* 40* 30* 10*          (idxred = 3, idxblue = 5, stacksize = 9)
1** 3** 4** 5** 6** 60* 50* 40* 30* 10*          (idxred = 4, idxblue = 5, stacksize = 10)
Clearing out both stacks...
*** *** *** *** *** 60* 50* 40* 30* 10*          (idxred = -1, idxblue = 5, stacksize = 5)
*** *** *** *** *** *** *** *** *** ***          (idxred = -1, idxblue = 10, stacksize = 0)
```

Assignment is on the next page.

Assignment 2 – Kyle Guarco

Driver.java

```
public class Driver
{
	public static void main(String[] args)
	{
		DoubleStack<Integer> ds = new DoubleStack<>(10);
		ds.redPush(1);
		ds.bluePush(10);
		ds.print();
		ds.redPush(2);
		ds.bluePush(20);
		ds.print();
		System.out.println("Popping from the red stack: " +
ds.redPop());
		ds.print();
		System.out.println("Popping from the blue stack: " +
ds.bluePop());
		ds.print();
		ds.redPush(3);
		ds.bluePush(30);
		System.out.println("Peeking at the blue stack: " +
ds.blueTop());
		ds.print();

		try {
			for (int i = 4; i < 7; i++)
			{
				ds.bluePush(i*10);
				ds.print();
				ds.redPush(i);
				ds.print();
			}
		} catch (IllegalStateException e) {
			System.out.println("Cannot add any more elements to the
stack. It's full!");
		}

		System.out.println("Clearing out both stacks ... ");
		ds.redClear();
		ds.print();
		ds.blueClear();
		ds.print();
	}
}
```

Assignment 2 – Kyle Guarco

DoubleStack.java

```
public class DoubleStack<E>
{
	// The character used when printing the stack on a null element
	private static final String DEBUG_NULL_ELEMENT = "*";

	/** The double stack. */
	private E[] stack;
	// The indicies of each stack.
	private int idxred, idxblue;

	@SuppressWarnings("unchecked")
	public DoubleStack(int capacity)
	{
		this.stack = (E[]) new Object[capacity + 1];
		this.idxred = -1;
		this.idxblue = capacity;
		print();
	}

	/**
	 * @param element
	 * @throws IllegalStateException
	 *    If the stack is full
	 */
	public void redPush(E element) throws IllegalStateException
	{
		if (isFull())
		{
			throw new IllegalStateException(
					String.format("Cannot push element %s because
the stack is full.",
					element.toString()));
		}

		idxred++;
		stack[idxred] = element;
	}

	/**
	 * @param element
	 * @throws IllegalStateException
	 *    If the stack is full
	 */
```

Assignment 2 – Kyle Guarco

```
	public void bluePush(E element) throws IllegalStateException
	{
		if (isFull())
		{
			throw new IllegalStateException(
					String.format("Cannot push element %s because
the stack is full.",
					element.toString()));
		}

		idxblue--;
		stack[idxblue] = element;
	}

	/**
	 * Removes the top element of the red stack and returns it.
	 * @return The top element of the red stack (null if none exist)
	 */
	public E redPop()
	{
		if (isEmpty() || redSize() == 0)
		{
			System.err.println("Cannot pop an element from the red
stack.");
			return null;
		}

		E element = stack[idxred];
		stack[idxred] = null;
		idxred--;
		return element;
	}

	/**
	 * Removes the top element of the blue stack and returns it.
	 * @return The top element of the blue stack (null if none exist)
	 */
	public E bluePop()
	{
		if (isEmpty() || blueSize() == 0)
		{
			System.err.println("Cannot pop an element from the blue
stack.");
			return null;
		}
```

```
            E element = stack[idxblue];
            stack[idxblue] = null;
            idxblue++;
            return element;
      }

      /**
       * Returns the top element of the red stack, but does not remove it.
       * @return The top element of the red stack (null if none exist)
       */
      public E redTop()
      {
            if (isEmpty())
            {
                  System.err.println("There's no element to peek at on the
red stack.");
                  return null;
            }

            return stack[idxred];
      }

      /**
       * Returns the top element of the blue stack, but does not remove it.
       * @return The top element of the blue stack (null if none exist)
       */
      public E blueTop()
      {
            if (isEmpty())
            {
                  System.err.println("There's no element to peek at on the
blue stack.");
                  return null;
            }

            return stack[idxblue];
      }

      public void redClear()
      {
            while (redSize() > 0)
                  redPop();
      }
```

Assignment 2 – Kyle Guarco

```
	public void blueClear()
	{
		while (blueSize() > 0)
			bluePop();
	}

	public int redSize()
	{
		return idxred + 1;
	}

	public int blueSize()
	{
		return stack.length - idxblue - 1;
	}

	public boolean isEmpty()
	{
		return redSize() == 0 && blueSize() == 0;
	}

	public boolean isFull()
	{
		return redSize() + blueSize() == stack.length - 1;
	}

	public void print()
	{
		String element = "";
		int pad = 0;
		for (int i = 0; i < stack.length - 1; i++)
		{
			// If the element is null, print the null character
instead
			element = stack[i] == null ? DEBUG_NULL_ELEMENT :
stack[i].toString();
			// String.repeat is used here for even padding (tab
doesn't look good otherwise)
			pad = 3 - element.length() > 0 ? 3 - element.length() :
0;
			System.out.print(element + DEBUG_NULL_ELEMENT.repeat(pad)
+ " ");
		}

		System.out.printf("\t(idxred = %d, idxblue = %d, stacksize =
```

Assignment 2 – Kyle Guarco

```
%d)\n",
                        idxred, idxblue, redSize() + blueSize());
      }
}
```