# Exercise 1

Aaron Patrick Monte
k20059926
Week 2 - Exercise

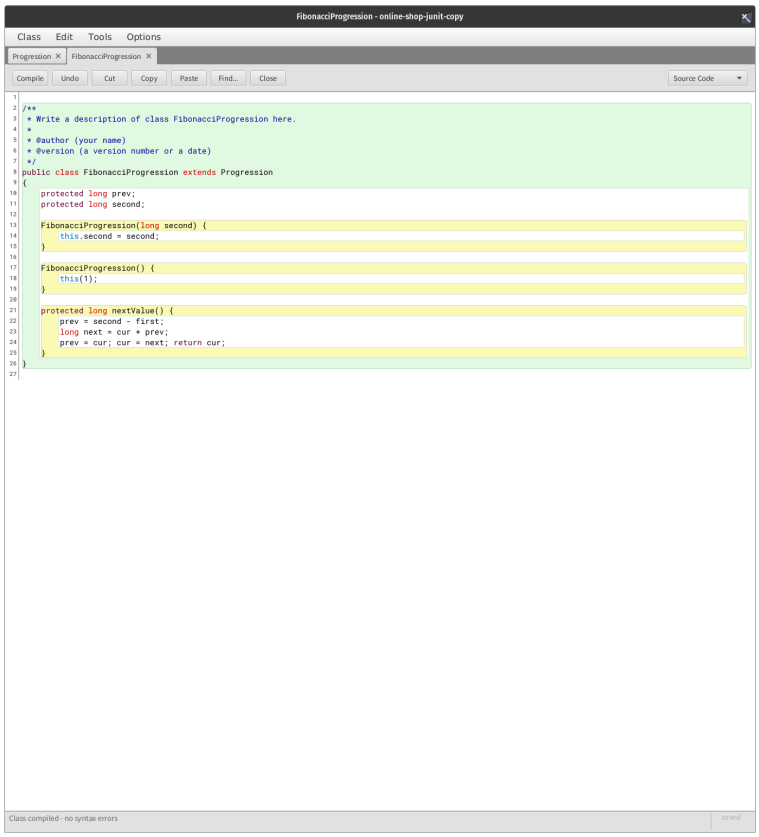
```
class TestProgression2 {
    public static void main(String[] args) {
        Progression prog;

        prog = new ArithProgression(5);
        prog.printProgression(5);
        prog.printProgression(7);

        prog = new GeomProgression(2);
        prog.printProgression(5);
        prog.printProgression(7);

        prog = new FibonacciProgression(3);
        prog.printProgression(5);
        prog.printProgression(7);
    }
}
```

```
>java TestProgression2
0 5 10 15 20
0 5 10 15 20 25 30
1 2 4 8 16
1 2 4 8 16 32 64
0 3 3 6 9
0 6 6 12 18 30 48
?
```

Explain why FibonacciProgression behaves differently than other subclasses.
Modify this class to achieve the expected behaviour.

When the second instance is called, the value for prev isn't reset.

Solution: define prev when the call is made, not in the constructor.




## Class Progression (1)

| Modifier | Class | Package | Subclass | World |
|---|---|---|---|---|
| public | Y | Y | Y | Y |
| protected | Y | Y | Y | N |
| no modifier | Y | Y | N | N |
| private | Y | N | N | N |

```
package progress;
public class Progression {
    protected long first;   // first value of the progression
    protected long cur;     // current value of the progression

    protected Progression() {          // default constructor
        cur = first = 0;
    }
    protected long firstValue() {  // resets the progression to the first value
        cur = first;    return cur;
    }
    protected long nextValue() {  // advances the progression to the next valu
        return ++cur;    // default next value
    }
```

## Class FibonacciProgression

$?, F_0, F_1, F_2, F_3, F_4, F_5, \ldots$

since we want:
second = first + prev

```
class FibonacciProgression extends Progression {
    // inherits fields:  first,  cur
    protected long prev;  // the previous value

    FibonacciProgression(long second) { // the second value given
        prev = second – first;          // default first value = 0
    }
    FibonacciProgression() {   // default constructor setting the second value to 1
        this(1);
    }

    //  inherits methods firstValue() and printProgression(int)
    protected long nextValue() {     // specialize nextValue() from Progression
        long next = cur + prev;
        prev = cur;   cur = next;   return cur;
    }
    public static void main(String[] args) { ... }   // test FibonacciProgression
}
```

prev + cur -> next
prev + cur -> next

# Exercise 2

- In class SLinkedList<E>, show Java code for methods:

```
// return the first element, but don't remove it from the list
public E elementAtHead() { ... }

public void insertAtHead( E newElem ) { ... }
public void insertAtTail( E newElem ) { ... }
public E removeAtHead() { ... }
```

SLinkedList - online-shop-junit-copy

Class Edit Tools Options

Node SLinkedList

Compile Undo Cut Copy Paste Find... Close Source Code

```java
/**
 * Write a description of class SLinkedList here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class SLinkedList<E>
{
    protected Node<E> head;
    protected Node<E> tail;
    protected long size;

    public SLinkedList()
    {
        head = null;
        tail = null;
        size = 0;
    }

    public E elementAtHead() {
        if (head == null) {
            System.out.println("Linked list is empty");
            return null;
        }
        else {
            return head.getElement();
        }
    }

    public void insertAtHead(E newElem) {
        Node<E> insert = new Node(newElem, head.getNext());
        head = insert;
    }

    public void insertAtTail(E newElem) {
        Node<E> insert = new Node(newElem, null);
        tail = insert;
    }

    public E removeAtHead() {
        head = head.getNext();
        return head.getElement();
    }
}
```

saved

# Exercise 3

Give code for method "contains" in this class:

```
public class SLinkedListExtended<E> extends SLinkedList<E> {
   // returns true if and only if, "element" is in the list
   public boolean contains(E element) { ... }

   public static void main(String[ ] args) {
        SLinkedListExtended<Integer> list =
                new SLinkedListExtended<Integer>();
        list.insertAtHead(2);   list.insertAtHead(4);  list.insertAtHead(6);
        System.out.println( "the list contains 4: " + list.contains(4));
        // prints: "the list contains 4: true "
}
```

```
public boolean contains(E element) {
        found = False;
        Node<E> current = head;
        while (current.getNext() != null)
                if (element == current.getElement()) {
                        found = True;
                }
                current = current.getNext();
        }
        return found;
}
```