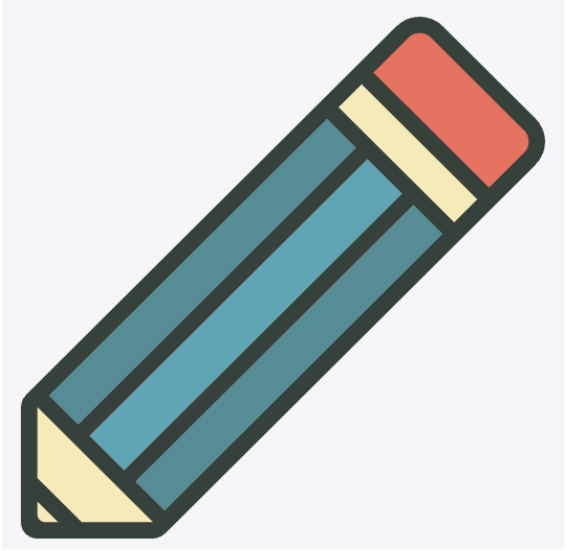
OOP REVIEW 2

Unplugged



With a partner but without digital technology, complete these activities.

They are intended to review basic OOP concepts such as:

- Encapsulation (including data hiding)
- Modifiers (public/private/static etc)
- General coding principles (iteration/selection/data types etc)

Your discussion may elicit questions about the concepts covered.

Note these questions or keep these questions in mind for further discussion.

A program models a PencilCase that stores multiple Pencil objects. Each Pencil stores the amount of lead remaining and can perform actions such as writing. The PencilCase can store up to 10 pencils and monitors their condition.

```
package pencilProject
public class Pencil {

    private int leadRemaining;

    public Pencil(int leadRemaining) {
        this.leadRemaining = leadRemaining;
    }

    public void write() {
        if (leadRemaining > 0) {
            leadRemaining -= 10;
            if (leadRemaining < 0) {
                leadRemaining = 0;
            }
            System.out.println("Writing... Lead remaining: " +
leadRemaining + "%");
        }
        else {
            System.out.println("This pencil has no lead left.");
        }
    }

    public boolean hasLittleLead() {
        return leadRemaining <= 20;
    }

    public int getLeadRemaining() {
        return leadRemaining;
    }
}
```

1. Identify an instance variable used in the Pencil class and describe its purpose.

[1]

2. State the output produced when the following code is executed:

```
Pencil p = new Pencil(15);
p.write();
p.write();
```

[2]

Here is the PencilCase class.

```
package pencilProject
public class PencilCase {

    private Pencil[] pencils;
    private static int pencilCount = 0;
    private static final int MAX_PENCILS = 10;

    public PencilCase() {
        pencils = new Pencil[MAX_PENCILS];
    }

    public void addPencil(Pencil p) {

        if (pencilCount < MAX_PENCILS) {
            pencils[pencilCount] = p;
            pencilCount++;
            System.out.println("Pencil added. Total pencils: " +
pencilCount);
        }
        else {
            System.out.println("PencilCase is full!");
        }
    }

    public void usePencils() {
```

```
        for (int i = 0; i < pencilCount; i++) {
            pencils[i].write();
        }

    }

    public void checkPencils() {

        int lowLeadCount = 0;

        for (int i = 0; i < pencilCount; i++) {
            if (pencils[i].hasLittleLead()) {
                lowLeadCount++;
            }
        }

        if (lowLeadCount > 5) {
            System.out.println("Lead is low!");
            refillCase();
        }
    }

    private void refillCase() {

        System.out.println("Empty pencil case, refill!");

        pencilCount = 0;

        for (int i = 0; i < MAX_PENCILS; i++) {
            pencils[i] = new Pencil(100);
            pencilCount++;
        }

        System.out.println("Refilled with 10 new pencils.");
    }

    public void displayPencils() {

        for (int i = 0; i < pencilCount; i++) {
            System.out.println("Pencil " + (i+1) + " lead: " +
pencils[i].getLeadRemaining() + "%");
        }

    }
}
```

3. Explain the purpose of the static modifier in:

```
private static int pencilCount;
```

[1]

4. What is the purpose of the checkPencils method?

[3]

```
package pencilProject
public class Main {

    public static void main(String[] args) {

        PencilCase myCase = new PencilCase();


        for (int i = 0; i < 10; i++) {
            Pencil p = new Pencil(100); //construct pencil outside case
            myCase.addPencil(p);
        }

        System.out.println("\nUsing pencils...\n");

        // simulate writing
        myCase.usePencils();
        myCase.usePencils();
        myCase.usePencils();
        myCase.usePencils();
        myCase.usePencils();
        myCase.usePencils();

        System.out.println("\nChecking pencil case...\n");
        myCase.checkPencils();

        System.out.println("\nCurrent pencil status:\n");
        myCase.displayPencils();
    }
}
```

5. In Main, Pencils are constructed externally before being added to the PencilCase.

   This means that they can exist independently and be used elsewhere in the program, not just in the PencilCase. What is the relationship between the PencilCase and the Pencil called?

   [1]

6. Complete the line of code that would change the relationship to *containment:*

   [1]

   myCase.addPencil(_____);

A new class ColoredPencil class is required.

A ColoredPencil:

- is a type of Pencil
- stores an additional private attribute color
- has a method draw() which reduces the lead by 5 units

7. Write the signature of the ColoredPencil class.

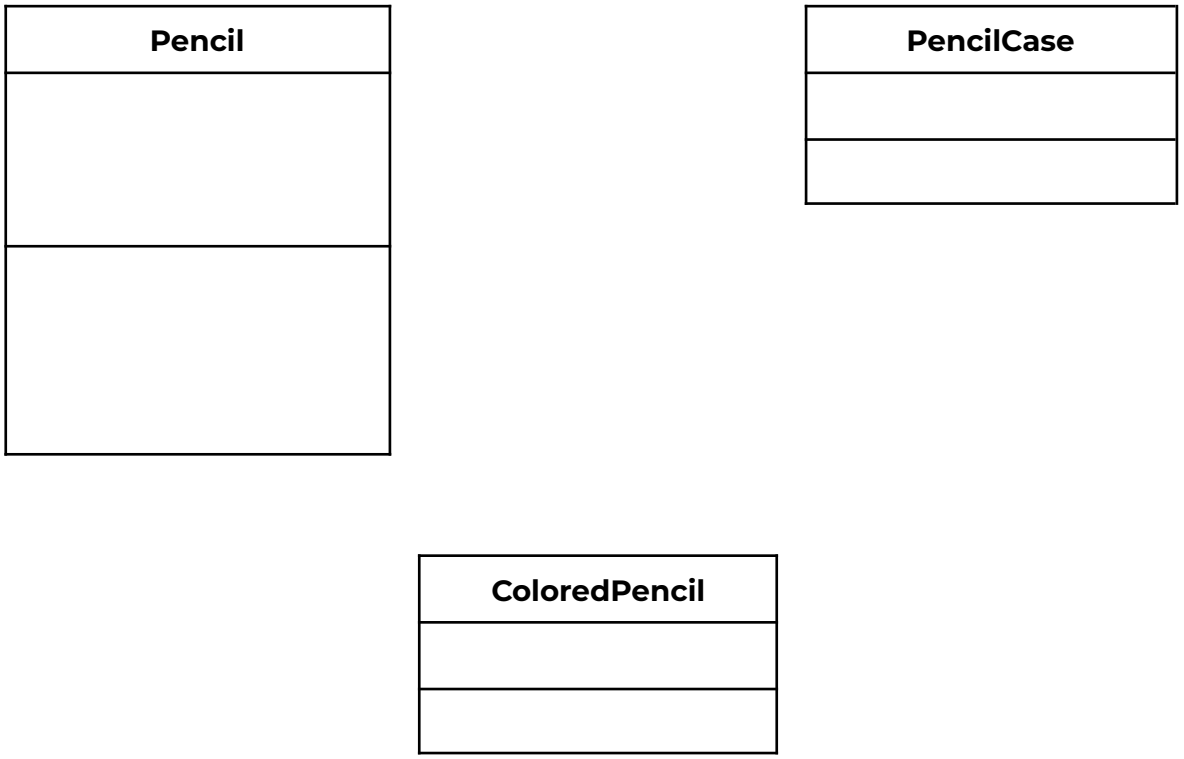
   [1]

8. Explain whether or not the following ColoredPencil class method will work as intended:

```
public void draw(){
      leadRemaining = leadRemaining - 5;
      if(leadRemaining<0){
            leadRemaining = 0;
      }
}
```

[2]

9. Complete the UML diagram of the Pencil class and show the relationship between the 3 classes.



HL Extension

a) A teacher keeps a LinkedList of Pencil objects representing pencils collected from students after an exam.

Each Pencil object has the following methods:

- public int getLeadRemaining()
- public void write()

A pencil is considered too short to use if its lead remaining is less than 10.

Write a method called:

```
public int removeShortPencils(LinkedList<Pencil> pencils)
```

[6]

b) A Node class is used to represent a linked list of Pencil objects.

Each Node contains:

- a reference to a Pencil
- a reference to the next Node in the list

```
public class Node{
    private Pencil data;
    private Node next;

    public Pencil getData()
    {
        return data;
    }

    public Node getNext(){
        return next;
    }
}
```

Write a method:

```
public int countShortPencils(Node head)
```

Which will:

- Check if the list is empty.
- Traverse the linked list starting from `head`.
- Count how many `Pencil` objects have leadRemaining < 10.
- Return the count.

[6]