Code Smells

CSC 210 Practice Exercises

The code below is attempting to manage a list of tasks. Each task has a description and information on whether the task has been completed or not.

What code smells do you see? How would you fix them?

```java
import java.util.ArrayList;

public class BadTaskManager {

    private ArrayList<String> descriptions;
    private ArrayList<Boolean> completenesses;

    public BadTaskManager() {
        descriptions = new ArrayList<String>();
        completenesses = new ArrayList<Boolean>();

    }

    public void addTask(String description, Boolean isCompleted) {
        descriptions.add(description);
        completenesses.add(isCompleted);
    }

    public void updateTask(int index,
                           String updatedDescription,
                           Boolean updatedCompleteness) {
        if (index >= 0 && index < descriptions.size()) {
            descriptions.set(index, updatedDescription);
            completenesses.set(index, updatedCompleteness);
        }

27          else System.out.println("Invalid index.");
28  
29      }
30  
31      public void DeleteTask(int index) {
32          if (index >= 0 && index < descriptions.size()) {
33              descriptions.remove(index);
34              completenesses.remove(index);
35          }
36          else System.out.println("Invalid index.");
37      }
38  
39      public void printTasks() {
40          for (int i = 0; i < descriptions.size(); i++) {
41              System.out.println(descriptions.get(i));
42              System.out.print("Task is ");
43              if (completenesses.get(i)) System.out.print("completed.");
44              else System.out.print("not completed.");
45          }
46      }
47  
48  }
```

Answer

- Primitive obsession (lines 5-6 and throughout). Solution: create a `Task` class with private instance variables, getters and setters;

```
public class Task {
    public String description;
    public boolean isCompleted;

    public Task(String description) {
        this.description = description;
    }

    public String getDescription() {
        return description;
    }

    public boolean isCompleted() {
        return isCompleted;
    }

    public void complete() {
      isCompleted = true;
    }
}
```

- Duplicate code (lines 22-27 and 32-36) to check if index is valid. Solution: create a method that checks if index is inbounds. The method returns a boolean.
- Long method for `printTasks()` – this could be simplified using polymorphism, overriding `toString()`. Solution: add `toString()` to `Task` class, rewrite `printTasks()` :

```
public void printTasks() {
  for (Task t : tasks) {
    System.out.println(t.ToString());
  }
}
```