```
/** A repeatable action to be performed by the app */
abstract class Action implements Repeatable{
    List<Task> tasks; //1 or more tasks which are parts of this action
    abstract void perform();
}
```

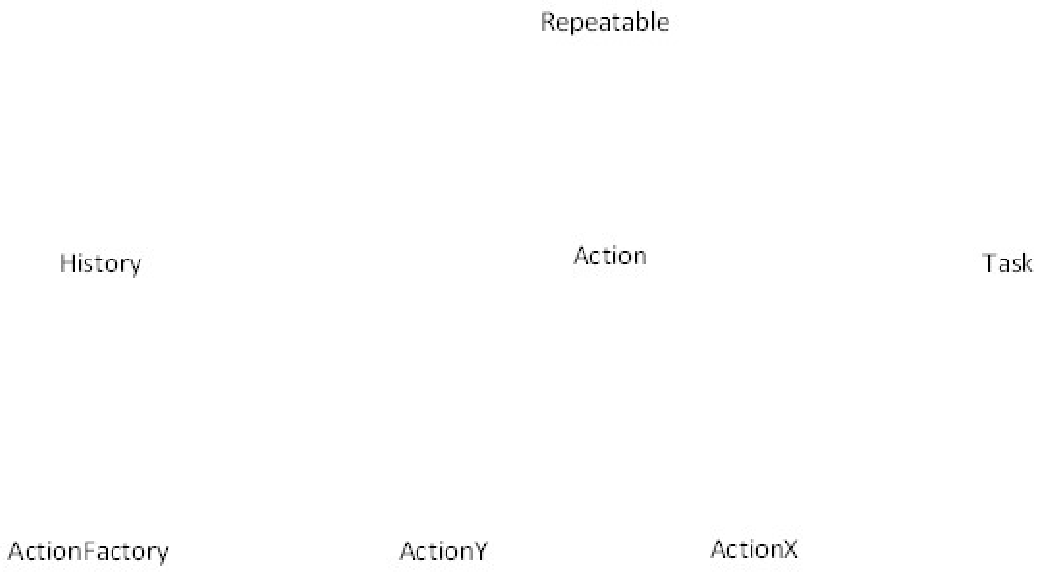
`ActionX, ActionY`: Inherits from the `Action` class.

`ActionFactory`: creates `ActionX, ActionY` objects and adds them to the `History` object.

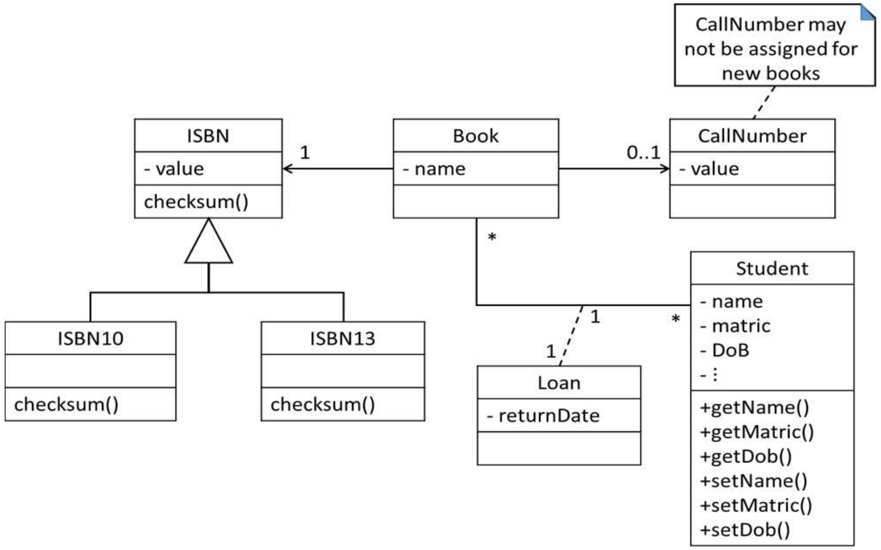
```
class Task{
    Repeatable owner; //the action that owns this task
    Task previous; // the previous task, if any

    Task(Repeatable owner, Task previous){
        //..
    }
}
```

```
interface Repeatable{
    void repeat();
}
```

```
class History{
    List<Repeatable> history;
    void add(Repeatable r){
        //..
    }
}
```



Consider the library management system (LMS) in the university implemented as shown in the class diagram below. Answer part (a) ~~and (b)~~ based on this class diagram



a. "Tarzan (a student) borrows a new copy of JungleBook with : ISBN13: 978-0123456789 and Beauty and the Beast with call number FC999c.1. JungleBook has to be returned within 1 week, by November 29, 2019. Beauty and the Beast, however, can be returned on December 23, 2019." Draw an appropriate UML diagram representing the state of the system after Tarzan borrows the books.