Ohjelmistotuotanto, kurssikoe 10.5.2016

Kirjoita jokaiseen palauttamaasi konseptiin kurssin nimi, kokeen päivämäärä, nimesi ja opiskelijanumerosi.

Vastaa tehtäviin ytimekkäästi. Minkään tehtävän vastauksen pituuden ei tule ylittää kolmea sivua, kaksikin sivua tiukkaa asiaa riittää hyvin.

Vastaa jokainen tehtävä omalle konseptilleen.

HUOM: älä viittaa vastauksissasi asioihin pelkästään luentomateriaalissa mainituja lyhenteitä/termejä käyttäen, selitä myös mitä asioilla tarkoitetaan.

1. (7p)
   (a) (2p) Mitä sisältyy ohjelmiston vaatimusmäärittelyyn?
   (b) (4p) Kerro mitä ketterään vaatimusmäärittelyyn liittyvillä termeillä User story ja Product backlog tarkoitetaan. Anna konkreettinen esimerkki oikeaoppisesta user storystä ja product backlogista.
   (c) (1p) Mikä on sprint backlog? Anna pieni esimerkki hyvästä sprint backlogista
2. 6p
   (a) Mitä tarkoitetaan ohjelmiston verifioinnilla ja validoinnilla?
   (b) Miten ohjelmiston validointi hoidetaan ketterissä menetelmissä
   (c) Mitä tarkoitetaan käsitteillä continuous integration ja continuous deployment
3. 7p
   (a) (2p) Ketterien menetelmien yhteydessä käytetään usein lähestymistapaa, josta käytetään englanninkielistä nimitystä incremental design tai incremental architecture, joskus myös evolutionary design. Selitä mitä käsitteellä tarkoitetaan. Mitä hyviä puolia ja mitä riskejä lähestymistavan noudattamisessa on?
   (b) (2p) Mitä sisäisen laadun kannalta ongelmallisia asioita tehtäväpaperin lopun koodissa on?
   (c) (3p) Selitä miten refaktoroisit tehtäväpaperin lopun koodia soveltaen suunnittelumalleja tai muita tilanteeseen sopivia ratkaisuja. Vastaukseen tulee liittää mukaan luonnosmaista koodia, pelkkä suunnittelumallien nimien luetteleminen ei riitä. Älä kuitenkaan kirjoita "javaboilerplatea" (luokka+näkyvyysmäärittelyt ym.) kokonaisuudessaan

Tehtävään 3 liittyvä ohjelmakoodi

```
public class UserStory {
    private int sprint;
    private String name;
    private int estimate;
    private String status;
    private List<String> tasks;

    public UserStory(String name, int sprint, int estimate, String status, List<String> tasks) {
        this.name = name;
        this.sprint = sprint;
        this.estimate = estimate;
        this.status = status;
        this.tasks = tasks;
    }

    public boolean done(){
       return status.equals("COMPLETED");
    }

    public void changeStatus(String status) {
        this.status = status;
    }

    public int estimate() {
        return estimate;
    }

    public String status() {
        return status;
    }

    public String name() {
        return name;
    }

}

public class BacklogReader {
    private String file;

    public BacklogReader(String file) {
        this.file = file;
    }

    public List<UserStory> readFromFile(){
        List<UserStory> items = new ArrayList<>();
        // read stories from file
        return items;
    }

    public void saveToFile(){
        // save stories to file
    }
}
```

```
public class Backlog {
    private List<UserStory> stories;
    private BacklogReader reader;

    public Backlog() {
        reader = new BacklogReader("tmc-cli");
        stories = reader.readFromFile();
    }

    public List<UserStory> completedStories() {
        ArrayList<UserStory> competed = new ArrayList<>();
        for (UserStory story : stories) {
            if ( story.done() ) {
                competed.add(story);
            }
        }
        return competed;
    }

    public List<UserStory> epics() {
        ArrayList<UserStory> epicStories = new ArrayList<>();
        for (UserStory story : stories) {
            if ( story.estimate()>20 ) {
                epicStories.add(story);
            }
        }
        return epicStories;
    }

    public String export(String format, boolean onlyRemaining) {
        String output = "";
        if ( format.equals("xml")){
            output = "<stories>";
        }
        for (UserStory story : stories) {
            if (onlyRemaining && !story.done() ) {
                continue;
            }
            if ( format.equals("text")) {
                output += story.name()+ " " + story.estimate()+ " " + story.status() + "\n";
            } else if ( format.equals("xml"))  {
                output += "<story><name>" + story.name() +
                            "</name><estimate>" + story.estimate() +
                            "</estimate><status>" + story.status() +
                          "</status><story>\n";
            } else {
                throw new IllegalArgumentException("format not supported");
            }
        }
        if ( format.equals("xml")){
            output += "</stories>";
        }

        return output;
    }

    // other methods...
}
```