Continuous Integration Tutorial

These tasks should be carried out by each team as a group on one computer:

1. Commit the sample code to the Git repository

2. Configure your project in Jenkins to:
    a. point at the sample code in your Git repository
    b. poll Git every 5 minutes
    c. e-mail team members in cases of build failures

3. Add a package called **com.examples.one.petstore**
4. Commit the code to SVN and confirm that Jenkins builds the code successfully

Each team should now split up into two groups, with each group working on one computer on different parts of the task at hand. Attempt each of the following tasks as a team but split up the work within each task between the two groups. Use a ***test-driven approach*** and commit/check out code on a regular basis often (when you finish a particular subtask).

Task 1 – Puppy Database for a Pet Store

Your customer, a pet store, requires a database to keep details about their puppies. The store never keeps more than 10 puppies in stock. If the user tries to add more than they are prompted with an error. In your *petstore* package, create two classes:

1. **Puppy**
    a. Properties
        i. id : int
        ii. name : String
        iii. breed : String (one of “Labrador”, “Doberman”, “Pitbull”)
        iv. loudBark : boolean
        v. exerciseRequiredPerDay : 0 >= x <= 4
        vi. friendlyWithChildren: Boolean
    b. Methods
        i. Constructor taking an id, name and breed
        ii. Getters and setters
2. **PuppyDB**
    a. Properties
        i. Puppies: ArrayList<Puppy>
    b. Methods
        i. addPuppy(Puppy) : boolean
        ii. delPuppy(id) : boolean
        iii. getPuppy(id) : Puppy

            iv. countPuppies() : int

Task 2 – Kitten Database for a Pet Store

Your customer also requires a database to keep details about their puppies. The store never keeps more than 5 kittens in stock. If the user tries to add more than they are prompted with an error. In your *petstore* package, create two classes:

1. **Kitten**
    a. Properties
        i. id : int
        ii. name : String
        iii. breed : String (one of "Persian", "Siamese", "Bengal")
        iv. sensitiveToFeeding: Boolean
    b. Methods
        i. Constructor taking an id, name and breed
        ii. Getters and setters
2. **KittenDB**
    a. Properties
        i. Kittens: ArrayList<Kitten>
    b. Methods
        i. addKitten(Kitten) : boolean
        ii. delKitten(id) : boolean
        iii. getKitten(id) : Puppy
        iv. countKittens() : int

Task 3 – Merged database

Your customer now wants to merge the two databases into one. Group common functionality between **Puppy** and **Kitten** into a superclass called **Pet**. Replace PuppyDB and KittenDB with PetDB:

1. PetDB
    a. Properties
        i. Pets: ArrayList<Puppy>
    b. Methods
        i. addPet(Pet)
        ii. delPet(Pet)
        iii. getPet(Pet)
        iv. countPets()
        v. countPuppies()
        vi. countKittens()
        vii. getRandomPuppy()
        viii. getRandomKitten()