**Problem Set 3, Part I**

**Problem 1: A class that needs your help**

**1-1)** *Revise the code found below:*

```
public class Participant {
    private String name;
    private int age;

    public Participant(String name, int age){
        setName(name);
        setAge(age);
    }

    public void setAge(int age){
        if(age<0){
            throw new IllegalArgumentException();
        }
        else {
            this.age=age;
        }

    }

    public void setName(String name){
        if(name==null){
            throw new IllegalArgumentException();
        }
        else{
            this.name=name;
        }
    }

    public String getName(){
        return this. name;
    }

    public int getAge(){
        return this.age;
    }
}
```

**1-2)**

```
a)   Participant qb = new Participant("Tom Brady", 40);

b)    qb.setAge(43);

c)   String mvp=qb.getName();
```

**Problem 2: Static vs. non-static**

**2-1)**

| **type and name of the variable** | **static or non-static?** | **purpose of the variable, and why it needs to be static or non-static** |
|---|---|---|
| int rank | non-static | stores the rank (1, 2, 3, ...) associated with a given Card object; needs to be non-static so every Card object will have its own instance of this variable |
| int cCount | static | Counts the amount of cards of suit Clubs (increments by one when a new club card is created). It will need to be static so it continues counting whenever an object is made. Each object will change the variable, and each object will share this variable therefore it must be static. |
| int dCount | static | Counts the amount of cards of suit diamond. Similar as cCount, it must be static to continue counting for each new card object created. Each object must be able to add onto the variable and not have their own independent version of it. Therefore each object will share the variable and it will need to be static. |
| int hCount | static | Counts the number of cards of the suit Heart. It must be static so each object of the card class can increment the variable, each object therefore shares this variable and it must be static |
| int sCount | static | Counts the number of cards of the spades suit. Because it counts each card, each object must share this variable without it changing. Rather, each card should only add to the variable therefore it must be static. |
| int rankAdd | Non static | This variable would add whatever the specified value is to the rank. This variable would be only subject to the object it is called upon therefore it is non static. |
| int suitAdd | Non static | This variable would add the specified value to the current suit number of the card object. Because it is used on an independent card object it would be non static. |

**2-2)**

a) **static or non-static?**: non static

**explanation:** the method would be called upon a card object to change its suit. Therefore in order to be called by an object, it needs to be  non static

b) **changes it would need to make**:
The setSuit method would have to change the instance variable that contained the suit of the card. Because setSuit takes in a string, it would have to change the suit variable which is also a string. It would then have to call the getSuitNum method and pass the string suit and store what is returned in the instance variable that stores in the int value for the suit, suitNum.

c) **example of calling it**:
```
c.setSuit(suit);
```

**2-3)**

a) **static or non-static?**:static
**explanation:** The method takes in an array of objects, it would not be called on an object itself. Therefore it would be static.

b) **example of calling it**:
```
int num =numFaceCards(myCards);
```

## Problem 3: Inheritance and polymorphism

**3-1)**

**The toString() method that Gee inherits is the default toString method from the object class. All classes extend the object class therefore it is a parent class of gee.**

**3-2)**

**x**
**a**
**c**
**d**

**3-3)**

| which println statement? | which method is called? | will the call compile (yes/no?) | if the call compiles, which version of the method will be called? |
|---|---|---|---|
| first one | why() | yes | the Tee version |
| second one | how() | yes | The zee version |
| third one | were() | yes | The gee version |
| fourth one | toString() | yes | The Tee version |
| fifth one | equals() | yes | The zee version |

**3-4)**

```
 public int total(){
    return this.get(e) + this.get(f) + this.get(c);

 }
```

**3-5)**

**a)Gee is a superclass of tee / tee is a subclass of gee therefore it is allowed**

**b)Tee is not a superclass of zee therefore it would not be allowed.**

**c)zee is not a superclass of yee / yee is not a subclass of zee therefore this is not allowed.**

**d) Object is the default parent class of all classes therefore it would be allowed**