Topic: Classes and Objects

Activity Guidelines

Group Size: 3

Method of Assigning Students: Count the number of students in the class, divide by 3, count off from 1 to the quotient, and group identical numbers.

Materials:

- ✓ Handout (one copy per group) with questions to be answered at the end of the session

Roles:

Coordinator/Leader: Clarifies goals and objectives, allocates roles for each team member and divides the tasks within the group.

Monitor/Evaluator: Person designed to evaluate the different ideas to approach the problem and make an accurate judgment of the most beneficial option.

Implementer: Person in charge to transform discussions and ideas into a technical solution for the given problem.

Individual Accountability: Every student will be working on teams for the exercises. Every team member is assigned a specific role that will allow everyone on the group to get involved and participate in the problem-solving activity. Every team member should participate in solving the exercises according to their determined role in the group.

Activity Summary

Practice skills on creation of classes, objects, arrays and methods. Assess the concept of in heritance in Java. The exercises require students to build a class with its respective fields, constructors, setters and getters in order to fully understand the properties of an object. Moreover, another class will be created which will become a subclass from the initial class students created. This is done with the purpose to demonstrate the concepts and importance of inheritance between classes and subclasses. Ultimately, they will need to test their classes using arrays and loops to produce the specified output.

INTRODUCTION TO COMPUTER SCIENCE
PEER SESSION

Classes and Objects

1. Add set and get methods to the following class:

```
public class Animal {

	private double weight;
	private double height;

	public Animal(double aheight, double aweight){
		weight = aweight;
		height = aheight;
	}

}
```

2. Create a subclass Dog for the class Animal:

3. Create a new class named Test, and will have to have the following:

   a. An array of size 3 of **Dog** objects
   b. A method that will return the index of the element of the previous array with the greatest weight
   c. A method that will return the index of the element with the smallest height

4. Create a main method that will perform calls to the previous methods and **trace** the execution.

Solution:

```
public class Animal {
	private double weight;
	private double height;

	public Animal(double aheight, double aweight){
		weight = aweight;
		height = aheight;
	}

	public void setWeight(double aweight){
		weight = aweight;
	}

	public void setHeight(double aheight){
		height = aheight;
	}

	public double getWeight(){
		return weight;
	}

	public double getHeight(){
		return height;
	}
}

public class Dog extends Animal {
	private String breed;
	private int age;

	public Dog(double aheight, double aweight, String abreed, int anage) {
		super(aheight, aweight);
		breed = abreed;
		age = anage;
	}

	public void setBreed(String abreed){
		breed = abreed;
	}

	public void setAge(int anage){
		age = anage;
	}

	public String getBreed(){
		return breed;
	}

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

```
public class Test {

	static int getIndexWeight(Dog[] d){
		int ind = 0;
		for(int i = 1; i < d.length; i++)
			if(d[ind].getWeight() < d[i].getWeight())
				ind = i;
		return ind;
	}

	static int getIndexHeight(Dog[] d){
		int ind = 0;
		for(int i = 1; i < d.length; i++)
			if(d[ind].getHeight() > d[i].getHeight())
				ind = i;
		return ind;
	}

	public static void main(String[] args) {
		Dog[] dogs = { new Dog(10, 9.5, "Chihuahua", 5), new Dog(45, 140.4,
		"Saint Bernard", 8), new Dog(30, 45.3, "German Shepperd", 10)};
		System.out.println("The greatest weight is at index: "+
			getIndexWeight(dogs));
		System.out.println("The smallest height is at index: "+
			getIndexHeight(dogs));
	}

}
```