Object Oriented Programming, Spring 2018 Midterm Exam Solution

1a)

```java
public class Student {
	private String name, id;
	private double cgpa;

	public static void main(String[] args){
		Student s1 = new Student("011162101", "Kratos", 3.0);
		Student s2 = new Student("011162102", "Thanos", 4.0);
		System.out.println("Before swap");
		System.out.println(s1);
		System.out.println(s2);
		swap(s1, s2);
		System.out.println("After swap");
		System.out.println(s1);
		System.out.println(s2);
	}

	public Student(String id, String name, double cgpa) {
		this.id = id;
		this.name = name;
		this.cgpa = cgpa;
	}
	public Student(Student s) {
		copy(s);
	}
	private void copy(Student s) {
		this.id = s.id;
		this.name = s.name;
		this.cgpa = s.cgpa;
	}
	private static void swap(Student s1, Student s2) {
		Student temp = new Student(s1);
		s1.copy(s2);
		s2.copy(temp);
	}
	public String toString() {
		return new String(id + " " + name + " " + cgpa);
	}
}
```

1a) OR

```java
class Animal {
	private boolean vegetarian;
	private String eats;
	private int noOfLegs;
	public Animal(boolean vegetarian, String eats, int noOfLegs) {
		this.vegetarian = vegetarian;
		this.eats = eats;
		this.noOfLegs = noOfLegs;
	}
	public void display() {
		System.out.println("Vegetarian: " + vegetarian);
		System.out.println("Eats: " + eats);
		System.out.println("No. of legs: " + noOfLegs);
	}
}

class Cat extends Animal {
	private String colour;
	public Cat(boolean vegetarian, String eats, int noOfLegs, String colour)
{
		super(vegetarian, eats, noOfLegs);
		this.colour = colour;
	}
	public void display() {
		super.display();
		System.out.println("Colour: " + colour);
	}
}

public class AnimalHeredityTest {
	public static void main(String[] args) {
		Cat c1 = new Cat(true, "Carrots", 4, "White");
		c1.display();
	}
}
```

1b)

1. The square brackets in the array argument are set after the variable name, it is not incorrect but it is discouraged in the Java programming reference.
2. The return value of *addTwoNumbers()* is ignored.

2a)

```
Output:
> $ 0
> $ 0
```

2a) OR

```
Output:
> $ 15.0Bazinga
> $ Bazinga105.0
> $ 105.0Bazinga
```

Sums of numerical values in *println()* are computed before an object in the parameter list is encountered, after they are only listed. Characters are treated for their ASCII values.

2b)

```
Output:
> $ Animal can be Omnivorous
> $ Dinosaur mostly Herbivorous
> $ but TRex is carnivorous
> $ Dinosaur mostly Herbivorous
> $ but TRex is carnivorous
> $ Dinosaur mostly Herbivorous
> $ but TRex is carnivorous
```

3)

(a) Packages help keep the source code organised into separate domain and can be differentiated by the namespace.
Packages can be found inside sub-packages and a low-level package without the correct access specifier does not gave the visibility of upper level classes.

(b) A class is an abstraction of data where members are encapsulated within it.
Reference is a variable that points to a memory allocation belonging to a class.
An object is a wrapper class with minimum set of interfaces, but undefined overall number of members, it is up to the developer to cast the object to the designated type for its task.

(c) Abstract classes are meant to be inherited, they cannot be initialised. A private abstract class would have not any visibility to the sub-class for its methods to be implemented and overridden.

4a)

```java
public class Main {

	public static void main(String args[]) {
		Pet[] pets = new Pet[2];
		pets[0] = new Cat(50);
		pets[1] = new Dog(60);

		for(Pet p: pets) {
			p.make_noise();
			System.out.println("Warm: " + p.is_heated());
			p.sleep();
		}
	}

}

Abstract public class Pet {
	int bodyTemperature;

	public Pet(int bodyTemperature) {
		this.bodyTemperature = bodyTemperature;
	}

	abstract void make_noise();
	boolean is_heated() {
```

```
                return (bodyTemperature >= 80);
        }
        final void sleep() {
                System.out.println("Pet sleeping");
        }
}

class Cat extends Pet {
        public Cat(int bodyTemperature) {
                super(bodyTemperature);
        }
        public void make_noise() {
                System.out.println("Meow");
        }
        boolean is_heated() {
                return (bodyTemperature >= 50);
        }
}

class Dog extends Pet {
        public Dog(int bodyTemperature) {
                super(bodyTemperature);
        }
        public void make_noise() {
                System.out.println("Ghew");
        }
}
```

4b)

To make the statement true when v1 and v2 are the same person, the name and the last name must be neglected in the equality check. The solution is to override the *toString()* method for the Voter class and return the voterId string and the age integer.

5a)

```
public class Vector2D() {
    double x, y;
    Vector2D() {
        x = 0;
        y = 0;
    }
    Vector2D(double x, double y){
        this.x = x;
        this.y = y;
    }
    Vector2D multiplication(double scalar) {
        return new Vector2D(scalar * x, scalar * y);
    }
    double multiplication(Vector2D vector) {
        double dotProduct = (x * vector.x) + (y * vector.y);
    }
}
```

5b)

a) In class B, ‘var’ cannot be incremented because is a constant, the final keywork needs to be removed before the type qualifier in class A. ‘var’ is also a static variable, any new subclasses will point to the only one integer variable, example ob1.var and ob2.var. If not programmatically necessary to share a common variable for all subclass of A, remove the static modifier.

b) In class C, the *void meth()* method should override the *void meth()* in class B, it is a final method so it cannot be overridden. The solution is to remove the keyword final before the method definition found in class B.