APS101- Midterm Sample Questions

Note: Java API descriptions similar to the following will be provided in the midterm.

Short Java API descriptions (all methods are public):

```
class Integer:
  Integer(int i) // An Integer with value i
  static int parseInt(String s) // = s's value, as an int.

class Double:
  Double(double d) // A Double with value d
  static double parseDouble(String s) // = s's value, as a double.

class String:
  String substring(int i, int j) // = the letters between i (inclusive) and j (non-inclusive).
  String substring(int i) // = the letters from i (inclusive) to the end.
  char charAt(int i) // = the character at index i (starting from 0)
  int indexOf(String s) // = the index of s in this String; -1 if s is not a substring.
  int indexOf(String s, int i) // = index of s in this String after index i; -1 if s not found.
  int length() // = the number of characters in this String.

class System:
    static PrintStream out //standard output stream
class PrintStream:
    print(String s)   //print string s to the output
    println(String s) //print string s to the output and terminates the line

class Math:
  min(int a, int b) // = the smaller of the given int values
  max(int a, int b) // = the larger of the given int values
  min(double a, double b) // = the smaller of the given double values
  max(double a, double b) // = the larger of the given double values
```

Question 1. [15 MARKS]

1. Given an `int` variable `price` and a `double` variable `discount`, write an expression that divides `price` by `discount` using integer division. Assume that `discount` is not zero.

2. Write an expression that evaluates to `true` if the variables `myKitchen` and `yourKitchen` are referencing the same object, and evaluates to `false` otherwise.

3. Given two `double` variables: `dog` and `cat`. Without using if-statements, write a single statement that returns `true` when `dog` has at least triple the value of `cat`, and returns `false` otherwise.

4. Write a method `calculatePrice`, which has two `double` parameters `basePrice` and `overHead` and returns their sum.

5. Write a class `Dog`, which has an instance variable that is a reference to an Owner object. It also has a constructor, which has one Owner parameter. Assume class `Owner` has already been defined.

6. Write a method called `getDimensions`, which has one parameter, a `JFrame`, and returns a `String` containing the `JFrame`s height and width, separated by a colon (:). Assume all necessary packages have been imported.

Question 2. [15 MARKS]

Write a class called `MoneyBag` in the space below, which stores money in terms of dollars and cents. The `MoneyBag` class has the following specifications:

Variable Type	Variable Description
`int`	The dollars contained in this `MoneyBag`.
`int`	The cents contained in this `MoneyBag`.
`double`	The largest amount stored in **any** `MoneyBag` so far.

Method	Method Description
`MoneyBag(int, int)`	The constructor for this `MoneyBag`, which takes in its dollar and cent amounts.
`add(MoneyBag)`	Adds the contents of this `MoneyBag` to the contents of the `MoneyBag` parameter. Returns the combined result in new `MoneyBag` object (as a `MoneyBag`).
`getRichest()`	Returns largest amount stored in a `MoneyBag` (as a `double`).

Some things to consider:

- The dollars parameter will be positive, and the cents parameter will be between 0 and 99.
- You may not use any if-statements.
- Comments are not necessary unless you feel that your code is not entirely clear.
- You should decide whether any of the variables or methods need to be made static.

Question 3. [15 MARKS]

Consider the following two classes.

```
/** A musician in a band. */
public class Musician {

  private String firstName;
  private String lastName;
  private double salary;
  public static int total;

  public Musician(String f, String l) {
    firstName = f;
    lastName = l;
    total++;
  }

  public void setSalary(double r) {
    salary = r;
  }
  public int getSalary() {
    return (int) salary;
  }
  public String getName() {
    return lastName + ',' + firstName;
  }
  public static int getTotal() {
    return total;
  }
  public static boolean isSameMusician(
    Musician m1, Musician m2) {
    return m1 == m2;
  }
  public String toString() {
   return getName() + ", " + salary;
  }
}
```

```
/** A band made up of three musicians.*/
public class Band {

  private Musician member1;
  private Musician member2;
  private Musician member3;
  private String name;
  private int salary;

  public Band(Musician m1, Musician m2,
    Musician m3) {
    member1 = m1;
    member2 = m2;
    member3 = m3;
    setSalary();
  }
  public Band() {
    member1 = new Musician("N1", "F1");
    member2 = new Musician("N2", "F2");
    member3 = new Musician("N3", "F3");
  }
  public Musician getMember1() {
    return member1;
  }
  public void setName(String n) {
    name = n;
  }
  public String getName() {
    return name;
  }
  public void setSalary() {
    salary = member1.getSalary()
    + member2.getSalary()
    + member3.getSalary();
  }
  public int getSalary() {
    return salary;
  }
}
```

On the next page is a `TestCase` subclass that tests these classes. In the space provided, for each `assertEquals` call write: **P** if the `assertEquals` passes, and **F** if the `assertEquals` fails. Assume all assert statements in each test are evaluated even if some early asserts fail.

To the right of each failure, write the actual value of the 2nd parameter in the space provided.

```
import junit.framework.TestCase;                                Result    If failed,

public class Tester extends TestCase {                          (P/F)     value returned
                                                                          by 2nd argument?

  public void test1() {
    Musician m = new Musician("N1", "F1");
    Band b = new Band();
    assertEquals(m.toString(), b.getMember1().toString());      ____   ____________________
  }

  public void test2() {
    int x = Musician.getTotal();
    Band b = new Band();
    assertEquals(x + 3, Musician.getTotal());                   ____   ____________________
  }

  public void test3() {
    Musician m1 = new Musician("N1", "F1");
    assertEquals("N1 F1", m1.getName());                        ____   ____________________


  public void test4() {
    Musician m1 = new Musician("AName", "ALastName");
    assertEquals("ALas", m1.getName().substring(1,4));          ____   ____________________
  }

  public void test5() {
    Musician m1 = new Musician("N2", "F2");
    m1.setSalary(40.99);
    Band b = new Band(m1, m1, m1);
    assertEquals(123, b.getSalary());                           ____   ____________________
  }

  public void test6() {
    Musician m1 = new Musician("N3", "F3");
    Musician m2 = new Musician("N3", "F3");
    assertEquals(true, Musician.isSameMusician(m1, m2));        ____   ____________________
  }

  public void test7() {
    Band b = new Band();
    assertEquals("", b.getName());                              ____   ____________________
  }

  public void test8() {
    Musician m1 = new Musician("N3", "F3");
    m1.setSalary(50);
    Band b = new Band();
    assertEquals(Integer.parseInt("0"), b.getSalary());         ____   ____________________
  }
}
```