**Name:**\_\_\_\_\_Ulvi Bajarani\_\_\_\_\_\_\_\_\_\_ SID **#:**\_\_\_20539914\_\_\_\_\_\_\_

1. Assume the following class definitions:

public class Clock {

private int second;

private int minute;

private int hour;

…

public void print ( ) {

System.out.printf (“Current time is: %d:%d:%d\n”,hour, minute, second);

}

@Override

public String toString( ) {

return String.format(“Current time is: %d:%d:%d\n”,hour, minute, second);

}

}

public class AlarmClock extends Clock {

private boolean alarmOn;

…

}

* 1. Create a constructor with parameters for AlarmClock class which is a child class of Clock. It initializes its parent class instance variables – second, minute, and hour – and initializes its own instance variable – a boolean – alarmOn.

**public AlarmClock(int hour, int minute, int second, boolean alarmDefine)**

**{**

**super(hour, minute, second);**

**alarmOn = alarmDefine;**

**}**

* 1. Write a print method for AlarmClock class which makes use of its parent class print method.

**public void printUseParent () {**

**super.print();**

**}**

* 1. Write a toString method for AlarmClock class which make use of its parent class toString method.

**@Override // *I’m not sure if it is needed, but it’s better to add.***

**public String toString()**

**{**

**return super.toString();**

**}**

1. For the following questions, assume that Student, Employee and Retired are all extended classes of Person, and all four classes have different implementations of the method getMoney. Consider the following code where … are the required parameters for the constructors:

Person p = new Person(…);

int m1 = p.getMoney( ); // assignment 1

p = new Student(…);

int m2 = p.getMoney( ); // assignment 2

if (m2 < 100000) p = new Employee(…);

else if (m1 > 50000) p = new Retired(…);

int m3 = p.getMoney( ); // assignment 3

* 1. The reference to getMoney( ) in assignment 1 is to the class
     1. **Person**
     2. Student
     3. Employee
     4. Retired
     5. none of the above, this cannot be determined by examining the code
  2. The reference to getMoney( ) in assignment 2 is to the class
     1. Person
     2. **Student**
     3. Employee
     4. Retired
     5. none of the above, this cannot be determined by examining the code
  3. The reference to getMoney( ) in assignment 3 is to the class
     1. Person
     2. Student
     3. Employee
     4. Retired
     5. **none of the above, this cannot be determined by examining the code**
  4. The relationship between Student class and Person class is referred to as a(n) \_\_\_\_ relationship.
     1. has-a
     2. **is-a**
     3. was-a
     4. instance-of
     5. alias

1. Given the following inheritance structure:

GoldenDelicious

MacIntosh

Fruit

Apple

Orange

* 1. Assume the following declarations:

Fruit fruit = new GoldenDelicious();

Orange orange = new Orange();

* + 1. Is the object fruit instanceof Orange? **\_\_NO\_\_\_**
    2. Is GoldenDelicious a direct subclass of Fruit? **\_\_NO\_\_\_**
    3. Is the object fruit instanceof Apple? **\_\_YES\_\_**
    4. Is object orange instanceof Fruit? **\_\_YES\_\_**
    5. Is object fruit instanceof MacIntosh? **\_\_NO\_\_\_**