CS211 Fall 2011

Dr. Kinga Dobolyi

Final Exam

Student Name: _____

Student G#: _____

Student signature for Honor Code:

_____

1. Complete the Person class below so that every **`Person`** object will ALWAYS have a name that is more than five characters long, and an age that is between 1 and 100 inclusive. (5 points)

```
public class Person{
      private String name;


      public Person(String nameIn){
```

2. Why is **`main`** a static method? (This can be answered in one phrase) (5 points)

3. How are polymorphism and dynamic binding related? (5 points)

4. List one benefit each, as we have discussed in class, of the following concepts (can be a phrase). Each benefit must be different from all of the others for full credit. (20 points):
    a. Static fields

    b. Primitive arrays

    c. Private methods

    d. Public fields

    e. Private fields

    f. Passing an argument by reference

    g. Constructors

    h. Interfaces

i. Inheritance

j. Abstract classes

5. Now list one drawback or "thing we have to be careful about otherwise it will cause a compilation/runtime error" each, as we have discussed in class, of the following concepts. Each answer must be appreciably different from the rest (20 points):
    a. Static fields

    b. Primitive arrays

    c. Private methods

    d. Public fields

    e. Private fields

f. Passing an argument by reference

g. Constrcutors

h. Interfaces

i. Inheritance

j. Abstract classes

6. Examine the three classes below (each in a different file):

```
import java.util.ArrayList;

public class Final1{
     public static void main(String[] args){


          ArrayList<Person> people = new ArrayList<Person>();
          people.add(new Person("Kinga"));
          people.add(new Student("David"));
          people.add(new GradStudent("John"));

          for(int i = 0; i < 3; i++){
               people.get(i).print();
               System.out.println("");
          }
```

```
    }
}
```

```
public class Person{
    private String name = "default";
    public Person(String n){ name = n; }
    public void print(){
        System.out.print("person :" + name);
    }
}
public class Student extends Person{
    public Student(String n){ super(n); }
    public void print(){
        System.out.print("student ");
        super.print();
    }
}
public class GradStudent extends Student{
    public GradStudent(String n){ super(n); }
    public void print(boolean argument){
        System.out.print("grad student ");
        super.print();
    }
}
```

a. What is the output of the code above? (6 points)

b. Change the code above to make **`people`** a list of objects. That is, remove the restriction that **`people`** can only store **`Person`** objects. Make sure the rest of the classes compile, after making this change (which might lead to more changes). (2 points)
c. Answer the following questions about short-circuiting in Java: (7 points)
   What is short-circuiting:

   An example of short-circuiting in an if-statement:

   Short circuiting is useful because:

7. Provide the output for the following code (18 points):

```
import java.util.ArrayList;
public class Final2{
     public static void main(String[] args){
          Object[] arrList = new Object[5];
          int[] intList = new int[5];
          ArrayList elt = new ArrayList();

          elt.add("hello");
          for(int i = 1; i < 6; i++)
               intList[i-1] = i;

          arrList[0] = intList;
          arrList[1] = elt;
          arrList[2] = "goodbye";
          arrList[3] = 3;
          arrList[4] = new Integer(3);

          for(int i = 0; i < 5; i++)
               System.out.println(intList[i]);
          for(int i = 1; i < 5; i++)
               System.out.println(arrList[i]);
          elt = changeMe(elt, intList, arrList);
          for(int i = 0; i < 5; i++)
               System.out.println(intList[i]);
          for(int i = 1; i < 5; i++)
               System.out.println(arrList[i]);
     }

     public static ArrayList changeMe(ArrayList elt,
               int[] intList, Object[] arrList){
          elt = new ArrayList();
          elt.add("David");
          intList[3] = 7;
          ((ArrayList)arrList[1]).add("John");
          arrList[2] = arrList[4];
          arrList[3] = 5;
          arrList[4] = new Integer(4);
          return elt;
     }
}
```

write output here:

8. Read through all of this question before beginning to write code. You are going to implement your own collection for this question that does not rely on arrays or an existing collection to store data. Your collection should be able to store objects of any type. You must always use good object-oriented design
    a. Write an interface that is called Sortable that has one method, **`getValue`**, which returns the value associated with a Sortable object. (4 points)

.

b. Next, write a class called **`Element`** that implements the **`Sortable`** interface. An element should store one object. An **`Element`** should link to two other **`Element`**s. Each element's value should be mapped to a primitive integer – how you want to do this is up to you, but you must guarantee that no two elements will have the same integer value, which will be returned by the **`getValue`** method. You MUST use at least one static variable in this class to implement part of the required functionality. (10 points)

c. Now, write a class called **`ElementCollection`**. It should have the following features (18 points):
    i. Elements should always be in sorted order, according to the result of **`getValue`**
    ii. The collection must have a method to return the "start" or "head" element
    iii. The collection must have a method to add an element (argument type must be **`Element`**, not **`Object`**)
    iv. The collection must have a method to return the size of the collection.
    v. The collection must have a single method to get either the "previous" or the "next" element in the collection. The method will decide which one to return based on an argument you pass to the method (you can pick an appropriate type for this argument). If there is no next or previous element, your method should throw an exception, **`NoException`**, which is a class in this same folder that is a direct child class of the class **`Exception`**.
    vi. You may not use any import statements (assume this file will be in the same folder as the other two).

Write on next page if needed