جامعة بيرزيت

BIRZEIT UNIVERSITY

Computer Science Department

Advanced Programming (COMP231)

Midterm Exam 20/12/2016

Student Name Ahmad Sawi

Student Number 1150007

Please choose you instructor

- ➢ [✓ Mr. Nael Qaraeen] Section 1
- ➢ Dr. Samer Zain
- ➢ Dr. Mamoun Nawahda
- ➢ Dr. Majdi Mafarja
- ➢ Dr. Nariman Amar

Question	Mark
1.	40
2.	16
3.	39
Bonus	5
Total	100

Question Number One (40 Points), Choose one best answer from the following:

1. Is there something wrong with the following program?

```
public class Test {
  public static void main(String[] args) {
    int list = new int[4];      list[0] ...
    for (int i = 0; i <= list.length(); i++) {
      sum += list[i];
    }
  }
}
```

A) The program has no errors.

B) The program has a syntax error.

[✓ C)] The program will produce a runtime error.

D) None of above.

2. Given the declaration Circle x = new Circle(), which of the following statement is most accurate.

A) x contains an object of the Circle type.

B) You can assign an int value to x.

C) x contains an int value.

[✓ D)] x contains a reference to a Circle object.

3. Given the declaration Circle[] x = new Circle[10], which of the following statement is most accurate.

[✓ A)] x contains a reference to an array and each element in the array can hold a reference to a Circle object.

B) x contains a reference to an array and each element in the array can hold a Circle object.

C) x contains an array of ten objects of the Circle type.

D) x contains an array of ten int values.

4. Regarding the static and instance methods and attributes, which of the following statements are true?

A) A static method can access instance attribute. ×

[✓ B)] A static attribute can be accessed by static and instance methods. ✓

C) An instance method cannot access static attribute. ×

D) none of the above.

instancy access all
static only accesses
static

5. An overloaded method consists of

A) The same method name with different types of parameters

B) The same method name with different number of parameters

C) The same method name and same number and type of parameters with different return type

[✓ D)] Both (a) and (b) above

6. Can a sub class override a private method located at its super class?

it doesn't even see private

[✓ A)] No, private methods cannot be overridden.

B) Yes, but the method at the super class must be a static method. ×

C) Yes, but you need to include the @Override annotation. ×

D) None of the above

7. A subclass inherits ______________ from its superclass.

A) protected method ✓

B) private method ×

C) public method ✓

[✓ D)] A and C

E) B and C

8. **If we have a class that has a protected method, which of the following statements is most accurate?**

A) the protected method can be accessed only within sub-classes ~

B) the protected method can be only accessed within classes in same package. ~

[✓ C] The protected method can be accessed by sub-classes and classes in same package. ✓

D) The protected method can only be accessed within classes that are in different package. x

9. **Suppose that you have a class named Order that has some attributes and methods such as getOrderTotal(). Is there anything wrong in the following code?**

```
Object obj = new Order(); ✓
System.out.println("Order Total = " + obj.getOrderTotal()); ←
obj = null; ✓
```

fails at matching

A) The error here is at third sentence, you cannot set an object to null. ? +

B) There is nothing wrong here. x

C) The error here is that the getOrderTotal() has to return a string. X

[✓ D] None of above.

10. **Is there anything wrong with the following code?**

```
public class PartTimeEmployee extends Employee{
    @Override
    public String toString(){return "Part Time Employee";}
 }
 class Employee extends Person{
     @Override
     public String toString(){return "Employee";}
  }
  class Person{
      String name;
      public Person(String t){
          name = t;
      }
      @Override
      public String toString(){ return "Person";}
  }
```

← no-argument ✓ ~~[illegible]~~

← no no-argument constructor

Person ✗
↓
Employee ✓
↓
partTimeEmployee ✓

A) Yes, the PartTimeEmployee class does not have a default constructor. ✗

[✓ B)] Yes, the Person does not have a default constructor. ✓

C) Yes, the Employee class does not have a default constructor. ✗ it has

D) All of above.

Write Your Answers Below

1.	2.	3.	4.	5.	6.	7.	8.	9.	10
C	D	A	B	D	A	D	C	D	B

-20

40

Question Number Two(20 Points)

What will be printed on screen after running the following code?

Part A)

```
class Grandfather {
    static String name = "Grand ";
    String doStuff() {
        return "GRAND_F ";
    }}
class Father extends Grandfather {
    static String name = "Father ";
    String doStuff() {
        return "FATHER ";
    }}
public class Child extends Father {
    static String name = "CH ";
    String doStuff() {
        return "CHILD ";
    }
    public static void main(String[] args) {
        Father f = new Father();
        System.out.println(((Grandfather) f).name +
                ((Grandfather) f).doStuff());
        Child c = new Child();
        System.out.println(((Grandfather) c).name +
                + ((Grandfather) c).doStuff() +
                ((Father) c).doStuff());
}}
```

overriding

Mat[illegible] net binding

Output

~~GrandGRAND_F~~

~~Father~~FATHER

~~CH~~CHILDCHILD

-4

Part B)

```
public class Test3 {
    public static void main(String[] args) {
        Object[] o = {new A(), new B(), new A()};
        System.out.println(o[0]);
        System.out.println(o[1]);
        System.out.println(o[2]);
    }
}
 class A extends B {
   public String toString() {
     return super.toString() + ">>" +  "F_Object";
   }
 }
 class B {
   public String toString() {
     return "S_Object";
   }
}
```

Output

S_Object>>F_Object

S_Object

S_Object>>F_Object

-0

16

Question Number Three, 40 Points)

A) Write a Java class called Security that has one static method. The static method is named encrypt(String str) that receives a string and returns it in encrypted format according to the following steps: (15 Points)

1. First we need to make sure that the string has only alphabets. If the string has one character that is not alphabet the method returns null.
2. Then string alphabets are all converted to lower case.
3. Next the string is reversed.
4. After that some of the characters are converted according to following rules
   a. o is replaced by 0 (Zero)
   b. f is replaced by #
   c. s is replaced by $
   d. and x is replaced by *

20

```
Public class Security{
    public static String encrypt(String str){
        ~~if(str~~
        Char[] arr = str.toCharArray(); boolean flag = true;
        for(int i=0; i< arr.length ; i++){
            ~~if( arr[i]~~
            if(!(Character.isLetter(arr[i]))){
            } return (null); -> flag = False;
        }

            ~~Str.toLowerCase~~ if(flag) {
            String LCStr = Str.toLowerCase();
            String reversed = reverser(LCStr);
            ~~String[] toReplace = { "o" , "f" , "s" , "x" };~~
            ~~String [] ReplaceTo = { "0" , "#" , "$" , "*" };~~
             ~~for(int i=0 ; i< reversed.length() ; i++){~~

             String replaced = replacer(reversed);
             return (replaced);
        }
        else
           return(null);     // so compiler doesn't give no return error
    }
```

```
private ~~public~~ static String reverser(String s){
        char[] arr = s.toCharArray();
        ~~for(i~~
        char[] rev = new Char[S.length()];
        for(i=0; i< arr.length; i++){
            rev[(arr.length)- i] = arr[i];
        }                                              ✓

        String reversed = new String(rev);
        return (reversed);
}

private ~~public~~ static String replacer (String s){
            ~~S.replaceAll("O"~~ String rep;
    rep = S.replaceAll('O', '0');
      rep = S.replaceAll('f', '#');
      rep = S.replaceAll('s', '$');                  ✓
      rep = S.replaceAll('x', '*');
       return(rep);
    }
}
```

B) Suppose that we have a class named Employee as shown at below UML diagram: (20 Points)

```
Employee
 - empId: int
 - name: String
 - address: String
 - basicSalary: double

+ Employee()
+  Employee(id,    name,
address, basicSalary)
//getter & setter
+ toString(): String
+ equals(Object):
boolean
```

Assume that the class is already implemented.

Now, create a method named **sortEmployees(Employee[])** that receives an array of employees and sort them in ascending order based on basic salary. The method should **return** an **ArrayList** of the sorted employees.

19

```
     import java.Lang.*; import java.util.*;
public ArrayList< > sortEmployees(Employee[] e ✓) {
     ~~Array~~ Employee temp;
     for(int i=0 ;(i< e.length)-1; i++){✓
            for(int k= i+1 ; k< (e.length) ; k++){✓
                 if((e[i].getBasicSalary()) >(e[k].getBasicSalary()) ✓ ✓
                      temp = e[i];
                      e[i] = e[K];
                      e[k] = temp; ✓
            }          }
     }
   ArrayList<Employee> List = new ArrayList<>(Arrays.asList(e));
   return(List);
}
```

C) (BONUS QUESTION) Suppose that the Employee class at B) was extended by three sub-classes SalesEmployee, PartTimeEmployee and Manager. (5 Points)

Assume that these classes are also already implemented.

Now, create a method named **processEmployees()** that receives an **ArrayList** that holds references to collection of objects of the three sub-classes. The method will invoke and print on screen the toString() method only if the object is of type **Manager**.

```
public static void processEmployees( ArrayList a ){
        for( int i=0; i< a.size() ; i++){
                if((a.get(i)) instanceof Manager)
                        System.out.println( (a.get(i)).toString()); //no need for typeca
        }
}
```

5

Some of the String methods are available below

```
String
-----------------------------------------------
+ String()
+ String(s : String)
+ String(b : StringBuffer)
+ String( : char [ ] c)
+ charAt(i : int) : char
+ compareTo(s : String) : int
+ concat(s : String) : String
+ endsWith(s : String) : boolean
+ equals(o : Object) : boolean
+ equalsIgnoreCase(o : String) : boolean
+ getChars(srcb : int, srce : int,  : char [ ] dst, dstb : int) : void
+ indexOf(s : String) : int
+ length() : int
+ replace(alt : char, neu : char) : String
+ startsWith(s : String) : boolean
+ substring(von : int) : String
+ substring(von : int, bis : int) : String
+ toCharArray() : char [ ]
+ toLowerCase() : String
+ toUpperCase() : String
+ trim() : String
+ valueOf() : String
```