# United International University
## Department of Computer Science and Engineering

CSI 211: Object Oriented Programming  Mid Exam : Summer 2018

Total Marks: 30  Time: 1 hour 45 minutes

---

*There are THREE questions. Figures in the right-hand margin indicate full marks.*

1. (a) Write a code fragment to create the following multidimensional integer array. (2)

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | |
| 8 | 9 | | |
| 10 | | | |

(b) Indicate true or false by T/F. (2)

i. An abstract class must have at least one abstract method.

ii. An abstract class can have concrete methods.

iii. In java a class cannot implement multiple interfaces.

iv. In java a class cannot implement multiple abstract classes.

(c) Underline the error, and write the fixed code. You CANNOT add/delete any line. The fixed code should produce the following output: **Hello World** (3)

```
abstract class A
{
  A(){
      System.out.print("Hello ");
  }
}
class B extends A
{
  B(){
    System.out.println("World");
  }
}
```

```
public class HelloWorld
{
  public static void main(String[]
  args)
  {
    A a = new A();
    a = new B();
  }
}
```

(d) Explain what is wrong with this code. Fix the code. You CANNOT add/delete any line. (3)

```
interface ABC {
     String s = "123";
     void method1(int a);
     void greeter();
}
class C implements ABC{
     String s = "new";

     public void greeter() {
          System.out.println("Hi");
     }
}
```

2. (a) How many objects are eligible for garbage collection after executing the following code (show graphically)?(3)

```
public class Cookie {
    String name;
    Cookie(String n)
    {
        name=n;
    }
    public static void main(String[] args) {
        Cookie c1=new Cookie("W");
        Cookie c2=new Cookie("X");
        Cookie c3=new Cookie("Y");
        Cookie c4=new Cookie("z");
        c1=c2;
        c1=c3;
        c2=c4;
    }
}
```

(b) Write a complete **Product** class implementing encapsulation. The class has the following instance variables: **name, price**. Use getter and setter methods in the class to set and get the values of the fields. However, ensure that name and price cannot be changed from outside of the class. Write a **constructor** to initialize the Product object with name only. (3)

(c) Write the output of the code. (4)

```
public class Test {
    static int a = 1;
    int b = 2;
    int c;
    void update(Test obj)
    {
        this.a++;
        b++;
        obj.c++;
    }
}
```

```
public class TestMain {
    public static void main(String[] args)
    {
        Test obj1 = new Test();
        System.out.println("obj1.a:"+obj1.a);
        System.out.println("obj1.b:"+obj1.b);
        System.out.println("obj1.c:"+obj1.c);
        Test obj2 = new Test();
        obj2.update(obj1);
        System.out.println("obj1.a:"+obj1.a);
        System.out.println("obj1.b:"+obj1.b);
        System.out.println("obj1.c:"+obj1.c);
        obj2=obj1;
        obj2.update(obj1);
        System.out.println("obj2.a:"+obj2.a);
        System.out.println("obj2.b:"+obj2.b);
        System.out.println("obj2.c:"+obj2.c);
    }
}
```

3. (a) Suppose youre building a software for a private organization. Now, write a class named **Employee**. It has two attributes **name** and **salary** with types respectively String and Floating point number. The constructor of Employee class initializes name and salary with this reference keyword. There is one method named **void printSalary()**. The classes that extend Employee are **PlatinumEmployee** and **SilverEmployee**. **PlatinumEmployee** class overrides **printSalary()** method by invoking parent method and also prints the name and bonus amount which is 15% of the actual salary and finally prints the total salary by adding the bonus amount. **SilverEmployee** class also overrides **printSalary()** in similar way except that the bonus amount is 7% in this case. Write the code of these two classes also. (4)

(b) Write the output of the code. (3)

```
public class A {
    A()
    {
        System.out.println("Inside A");
    }
    A(String msg)
    {
        System.out.println("A: "+msg);
    }

}
public class B extends A{
    B()
    {
        System.out.println("Inside B");
    }
    B(String msg)
    {
        System.out.println("B: "+msg);
    }
}
```

```
public class C extends B{
    C()
    {
        System.out.println("Inside C");
    }
    C(String msg)
    {
        System.out.println("C: "+msg);
    }
    public static void main(String[] args)
{
        C c1=new C();
        C c2=new C("University");
    }

}
```

(c) Write the output of the code. (3)

```
public class Animal {
    Animal()
    {
        System.out.println("Animal created");
    }
    void eat()
    {
        System.out.println("Animal eats");
    }
    void fly()
    {
        System.out.println("Animal fly");
    }

}
```

```
public class Bird extends Animal{
    Bird()
    {
        System.out.println("Bird created");
    }
    void fly()
    {
        System.out.println("Birds fly");
    }
    public static void main(String[] args)
    {
        Animal a= new Bird();
        a.fly();
        a.eat();
    }

}
```