Name:_____

ICS4U0 – Test - Objects

Application (32 marks) **27/32**

An iphone XS is available with a 5.8 – inch display starting at $1379 and a 6.5 – inch display starting at $1519. It comes in three colours: silver, space grey, and gold. The phone comes with 64 GB of memory. An upgrade to 256 GB is $210 and an upgrade to 512 GB is $480. AppleCare is available for $249.

1. In Java, define and create a class definition of type *iPhoneXS* based on the description above. Make sure to include the appropriate public/private keywords and variable types. Include all necessary fields (there should be five): (A – 5 marks) **5 /5**

```
public class
{



}
class Iphone {
private double size;
private double price;
private String colour;
private int memory;
public boolean AppleCare; .Excellent answers but the part below is not required.

public int Upgrade (int memory){
if (memory ==256){
price = price + 210;
}
else if (memory==512){
price = price+480;
}
}
}
```

2. Constructors

Name:_____

ICS4U0 – Test - Objects

a. Create a default constructor (meaning no parameters) that will create a silver iPhone that is as cheap as possible. We'll consider this the "base" model. (2 marks) **2 - minor mistake**

```
public Iphone(){
this.price = 1519; .Should be 1379
this.colour = "silver";
this.size = 5.8;
this.memory = 64;
this.Applecare = 0;
//no applecare because cheapest price

}
```

b. Create a constructor that receives size, colour, and memory. This constructor will create an iPhone. AppleCare will not be selected for this iPhone. (3 marks) **1**

```
public Iphone (double size; String colour; int memory){
this.size = size;
this.colour = colour;
this.memory = memory;
    .Need to initialize apple care and figure out the price.  All instance variables
    need to be set to a value.
}
```

c. Create a constructor that receives a parameter of type iPhone. This will create an object with the same values as the iPhone parameter passed in. (3 marks) **3**

```
public Iphone (Iphone iphone){
this.size = iphone.size
this.price = iphone.price;
this.colour = iphone.colour;
this.memory = iphone.memory;
this.AppleCare = iphone.AppleCare;

}
```

3. Create accessor methods for one instance field. You may assume that accessor methods have been created for the rest. (A – 1 marks) **1**

```
public int getPrice () { ← minor mistake
return price;
}
```

Name:_____

ICS4U0 – Test - Objects

4. Create the mutator methods for all NECESSARY FIELDS. Remember, you can upgrade OR downgrade any options (A – 10 marks) **.10/10 All situations are accounted for and the price is accurate.**

```
public void Fieldmutator (double newSize, String newColour; int newMemory,
boolean newAppleCare) {
this.price =1379; .Excellent job here setting it to the cheapest first and then
checking for upgrades.
if (newSize ==6.3) .1/1
    this.price =1519;

if (newMemory ==256)
    this.price = this.price+210;
}

if (newMemory ==180)
    this.price ==this.price+180

if (newAppleCare ==true){
this.price ==this.price+249
}

this.Size = newSize;
this.colour = newColour;
this.memory = newMemory;
```

5. Explain what class field will not need a mutator method? Explain why? (A- 2 marks) **.0**

The AppleCare will not need a mutator method as it is the same for all iphones, therefore it is not unique to a specific phone and remains the same for all iphones. **.Applecare adjusts the price so it needs to be handled through a mutator method.**

6. Create an instance method called PrintSpecs. This method will print all the specifications for the implicit object. (2 marks) **.2**

```
public void PrintSpecs () {
System.out.println (size);
System.out.println(price);
System.out.println(colour);
System.out.println(memory);
System.out.println(Applecare);
}
```

Name:_____

ICS4U0 – Test - Objects

7. Create a class method called ComparePrice that receives two parameters of type iPhone. This method will return the cheaper iPhone (or the first if it's the same). (3 marks) **2/3**

```
public static double Iphone (Iphone x, Iphone y){
if (x.price>y.price) ← missing (minor)
    return y.price;
else
   return x.price; .need to return the object and not the price.

}
```

Knowledge and Understanding (16 marks) .14/16

8. In your main program write the code to create the following objects. **Illustrate the objects that are created. If an error has occurred explain the error.**

   a. Create a base model iPhone called *x*. **.2/2**

   ```
   Iphone x = new Iphone ();
   ```

   b. Create a silver 6.5-inch iPhone called *y* with 256 GB. **.2/2**

   ```
   Iphone y = new Iphone (6.5, .”silver.”, 256);
   ```

   c. Create an iPhone called *z* that is a copy of the x. **.2/2**

   ```
   Iphone z = a; .should be x
   ```

   d. **Create a variable of type iPhone called *a*. Assign *a* to *z*. .2/2**

   ```
   Iphone a = z;
   ```

   e. Add Applecare to *z*. **.1/1**

   ```
   z.applecare.(); ← minor
   ```

f. Change the memory of *x* to 512 GB. **.1/1**

```
x.Fieldmutator(512) .You need
```

g. Change the price of *y* to $1519. **.0/2 → This cannot be done directly. Price should be available to change through a method.**

```
y.Fieldmutator(1519);
```

h. Change the colour of *a* to Space Grey. **.1/1**

```
a.Fieldmutator ("Space Grey");
```

9. Write the code that will determine and print the cheapest iPhone. (3 marks – A)

```
int low1 = ComparePrice(Iphone a,price, Iphone b.price);
int low2 = ComparePrice(Iphone z.price, Iphone b.price);
if (low1<low2)
  System.out.prnt;n(low1);
else{
System.out.println(low2)
}
```

Part 2:

TIPS (Level) 4- (85%) No additional feature present. The upgrade screen is nice but not all of the boxes connect and not all of them are explained fully. Good effort overall!

10. Based on the class iPhone (and its fields), create multiple menu screens that represent a Java main program that will make use of all the fields and methods. Provide various options in a graphical application and describe how your program will work and what you want it to do. Ensure that all constructors, instance methods and class methods are used. You do not need to write a program but you should explain each screen and where the various options take you. In addition, think of one additional feature to add to your application. It should be represented by a field (variable) and a new method in your program that gives the user a new option that enhances their experience overall when deciding to buy a new phone. It can be a service, new phone feature, or other option but it should give the user more options and/or flexibility. You can go to INSERT → NEW. Then you can click on the SHAPE option and provide elements that show what the menu screens will look like. There is also a text tool so you write the options directly into

Name:_____

ICS4U0 – Test - Objects

the various shapes. You can also import pictures but the screens should be easy to understand and all options must be accounted for.