Name:	QUID:

Some methods of the String, Character, and Integer classes

charAt(int index): char – String	Returns the char value at the specified index.
indexOf(int ch): int – String	Returns the index within this string of the first occurrence of the specified character.
indexOf(String str): int – String	Returns the index within this string of the first occurrence of the specified substring.
subString(int beginIndex): String – String	Returns a string that is a substring of this string.
subString(int beginIndex, int endIndex): String – String	Returns a string that is a substring of this string.
startsWith(String prefix): boolean – String	Tests if this string starts with the specified prefix.
endsWith(String suffix): boolean – String	Tests if this string ends with the specified suffix.
contains(String charSequence): boolean – String	Returns true if and only if this string contains the specified sequence of char values.
length(): int – String	Returns the length of this string.
isBlank(): boolean – String	Returns true if the string is empty or contains only white space codepoints, otherwise false.
isEmpty(): boolean – String	Returns true if, and only if, length() is 0.
strip(): String – String	Returns a string whose value is this string, with all leading and trailing white space removed.
trim(): String – String	Returns a string whose value is this string, with all leading and trailing space removed, where space is defined as any character whose codepoint is less than or equal to 'U+0020' (the space character).
toLowerCase(): String – String	Converts all of the characters in this String to lower case using the rules of the default locale.
toUpperCase(): String – String	Converts all of the characters in this String to upper case using the rules of the default locale.
static valueOf(double d): double - String	Returns the string representation of the double argument.
static valueOf(float d): float - String	Returns the string representation of the float argument.
static valueOf(int d): int - String	Returns the string representation of the int argument.
static valueOf(char d): char - String	Returns the string representation of the char argument.
static getNumericValue(char ch): int – Character	Returns the int value that the specified Unicode character represents.
static isDigit(char ch): boolean – Character	Determines if the specified character is a digit.
static isLetterOrDigit(char ch): boolean – Character	Determines if the specified character is a letter or digit.
static isLetter(char ch): boolean – Character	Determines if the specified character is a letter.
static isWhiteSpace(char ch): boolean – Character	Determines if the specified character is white space according to Java.
static isTitleCase(char ch): boolean – Character	Determines if the specified character is a titlecase character.
static isUpperCase(char ch): boolean – Character	Determines if the specified character is an uppercase character.
static isLowerCase(char ch): boolean – Character	Determines if the specified character is a lowercase character.
static parseInt(String str): int – Integer	Parses the string argument as a signed decimal integer.

1. Consider the class **Q1** given below in answering the questions that follow. Assume that the class **Q1** has already coded and ready to use getter and setter methods for all fields. Assume also that the ***clone*** and ***copy*** member methods as well as the class method ***copy*** are already coded and ready to use for this class.

```
1   package l02Assessment.Midterm;
2   public class Q1 {
3           private double x;
4           String y;
5           public boolean z;
6           public Q1(double x1, String y1) {
7                   x = x1;     y = y1;
8           }
9           public Q1(double x1, String y1, boolean z1) {
10                  x = x1;     y = y1;     z = z1;
11          }
12          private void m1() { }
13          public void m2(Q1 q) { }
14          public static void m3(double v) { }
15          private static void m4(Q1 q) { }
16          void m5() { }
17          @Override
18          public String toString() {
19                  return String.format("x=%06.2f, y=%s, z=%b", x, y, z);
20          }
21          public static void main(String[] args) {
22                  Q1 q1,q2,q3,q4,q5,q6,q7,q8,q9;
23                  q1 = new Q1(12.3412,"A");
24                  System.out.println("q1: "+q1);
25                  q2 = new Q1(123456.789,"BF",true);
26                  System.out.println("q2: "+q2);
27                  q2.x = 0;
28                  System.out.println("q2: "+q2);
29                  q3 = q1;
30                  q3.y = q2.y;
31                  System.out.println("q3: "+q3);
32                  System.out.println("q1: "+q1);
33                  q4 = q1; q4 = q3; q4 = q2;
34                  q5 = q1.clone();
35                  q5.z = true;
36                  System.out.println("q5: "+q5);
37                  System.out.println("q1: "+q1);
38                  q6 = q1.clone();
39                  q2.copy(q6);
40                  System.out.println("q1==q3: "+(q1==q3)+" q1==q6: "+(q1==q6));
41                  q6.y="XYZ";
42                  System.out.println("q6: "+q6);
43                  System.out.println("q2: "+q2);
44                  //Line 44
45          }
46  }
```

1. Consider the class **Q1** in answering the questions that follow. Assume that the class **Q1** has already coded and ready to use getter and setter methods for all fields. Assume also that the ***clone*** and ***copy*** member methods as well as the class method ***copy*** are already coded and ready to use for this class.

   A. **[2.5 Points]** Mark your answer as **X** in the True or False box. Consider each statement as independent of the other statements in this question.

Be aware that a correct answer earns **positive** credit, incorrect answer earn **negative** credit, and unanswered earns **zero** credit.

Inside	Statement	True	False
m1	The statement **double this.x=9.0;** is correct in syntax and stores 9.0 in the private member field **x** of the calling object.		**X**
m1	The statement **x=9.0;** is correct in syntax and stores 9.0 in the private member field **x** of the calling object.	**X**	
m1	The statement **setX(getX()+9.0);** is correct in syntax and adds 9.0 to the value of the private member field **x** of the calling object.	**X**	
m2	The statement **this.x=9.0;** is correct in syntax and stores 9.0 in the private member field **x** of the passed object **q**.		**X**
m2	The statement **this.setX(this.get(x)+9.0);** is correct in syntax and adds 9.0 to the value of the private member field **x** of the passed object **q**.		**X**
m3	The statement **setX(getX()+9.0);** is correct in syntax and adds 9.0 to the value of the private member field **x** of the calling object.	**X**	
m4	The statement **x=9.0;** is correct in syntax and stores 9.0 in the private member field **x** of the passed object **q**.		**X**
m4	The statement **this.setX(9.0);** is correct in syntax and stores 9.0 in the private member field **x** of the passed object **q**.		**X**
main	The statement **q1.copy(q3,q4);** is correct in syntax and copies q3 into q4.	**X**	
main	The statement **Q1.m4();** is correct in syntax and results in executing the body of the private method **m4**.		**X**
main	The statement **m1();** is correct in syntax and results in executing the body of the public method **m1**.		**X**

B. [2.5 Points] Briefly and direct to the point, yet adequately, answer each of the following questions:

I. What should you do to allow other classes to access the method **m5** of this class without having to create an object of this class?

Make m5 static

II. How many objects are in memory when the executing the code in the main reaches line 44?

Four (4)

III. What will you do if you are required to allow the creation of only specific objects of a certain type?

Use enum

IV. If you have a class with all of its methods and field being static. What should you do to prevent other classes from creating an object from this class?

Declare all of its constructors as private. If it has none, define in it a private empty constructor

V. What should you do to prevent the creation of more than one object from this class? Any attempts to create an object of this class should return a reference to the first created object from this class.

The constructors are all private.

Define a private object obj type of the class and initially it is null.

Define a public method such as getInstance to be called when object is needed from this class

> **If obj is null -> create a new object, assign its reference to obj**
>
> **Return obj**

2. **[10 POINTS]** Code the following:

   A. The enumeration type **Meal** to use in the next question. The enumeration type **Meal** has four objects **MEAL1**, **MEAL2**, **MEAL3**, and **MEAL4**. Each of these objects has the a description and price attributes as Cheeseburger 7 QR, Big Mac 14 QR, Big Tasty 16 QR, Big Tasty Mushroom 21 QR respectively.

   B. The class **Order** having the following:

Member fields. All of them are private.

- **number** is an integer representing the order number and it is assigned the value of **count** once an object of **Order** is created. First order have the **number** value **1**.
- **meal1Qnty**, **meal2Qnty**, and **meal3Qnty** are integers representing the quantity ordered from each meal type.
- **totPrice** is a float representing the total amount of ordered meals.
- **discount** is a float greater than or equal to zero and less than one. It represents the discount. Its value is set by the private member method ***setDiscount***.
- **voucherCode** is a string representing the code of the voucher to consider for discount. Initially is by default **null** of course.
- **totToPay** is a float representing the total amount to pay after deducting the discount.

Class fields

- **count** is public and it is an integer with an initial value of **0**. **count** keeps track of how many objects are created from this class.

Member methods

Assume that all **setters** and **getters** methods are already coded properly and ready to use, except for the ones you are asked to code in this question.

- ***useVoucher*** is public and if the code it receives is valid, it assigns it to **voucherCode** then calls the private method ***setDiscount***. A valid code must satisfy all of the following conditions: (1) code length is eleven characters, (2) first three characters are letters, (3) the first and third letters are capital while the second letter is small case, and (4) the last two characters are digits representing the discount percentage.
- ***setDiscount*** is private and is only called by ***useVoucher*** only when the **voucherCode** has been set to a non-null value. This method extracts the discount percentage from the member field **voucherCode** then divide it by 100 and assigns the result to the member field **discount**.
- ***setTotPrice*** is private and it calculates the total price of the meals without considering the discount and assigns the result of this calculation to the member field **totPrice**.
- ***setTotToPay*** is private and it calculates the total amount to pay after considering the discount and assigns the result of this calculation to the member field **totToPay**.

Class methods

- ***CompareTo*** is public and receives two **Order** objects. It returns **1** if the value of **totToPay** of the first **Order** object is greater than the value of **totToPay** of the second **Order** object, **-1** if less than, and **0** if both values are equal.

Constructors

- A constructor that does not receive any arguments. It must increment the class field **count** and assign a correct value to the member field **number**.
- A constructor that receives a voucher code. This constructor should make use of an existing constructor to increment the class field **count** and assign a correct value to the member field **number**, and then calls the member method ***useVoucher***.

```java
package l02Assessment.Midterm;
public enum Meal {
	MEAL1("Cheeseburger",7), MEAL2("Big Mac",14),
	MEAL3("Big Tasty",16), MEAL4("Big Tasty Mushroom",21);
	Meal(String description, int price){
		this.description=description; this.price=price;
	}
	String description;
	int price;
}
```

```java
package l02Assessment.Midterm;
public class Order {
	private int number, meal1Qnty, meal2Qnty,meal3Qnty,meal4Qnty;
	private float totPrice, discount, totToPay;
	private String voucherCode;
	private static int count=0;
	public Order() {
		++count;
		number=count;
	}
	public Order(String VoucherCode) {
		this();
		useVoucher(voucherCode);
	}
	private void setDiscount() {
		this.discount = Integer.parseInt(
				voucherCode.substring(voucherCode.length()-2))/100;
		/*not included in the question */
		setTotToPay();
	}
	private void setTotPrice() {
		totPrice=Meal.MEAL1.price*meal1Qnty+Meal.MEAL2.price*meal2Qnty+
				Meal.MEAL3.price*meal3Qnty+Meal.MEAL4.price*meal4Qnty;
	}
	private void setTotToPay() {
		totToPay = totPrice - totPrice*discount;
	}
	public static int CompareTo(Order o1, Order o2) {
		if(o1.totToPay>o2.totToPay) return 1;
		if(o1.totToPay<o2.totToPay) return -1;
		return 0;
	}
	public void useVoucher(String VoucherCode) {
		if(isValidVoucher(VoucherCode))
			setDiscount();
	}
	private boolean isValidVoucher(String code) {
		boolean c1=false, c2=false,c3=false,c4=false;
		if(code.length()==11) c1=true;
		if(Character.isLetter(code.charAt(0))&&
				Character.isLetter(code.charAt(1))&&
				Character.isLetter(code.charAt(2)))
			c2=true;
		if(Character.isUpperCase(code.charAt(0))&&
				Character.isLowerCase(code.charAt(1))&&
				Character.isUpperCase(code.charAt(2)))
			c3=true;
		if(Character.isDigit(code.charAt(code.length()-2))&&
				Character.isDigit(code.charAt(code.length()-1)))
			c4=true;
		return c1&&c2&&c3&&c4;
	}
	public int getMeal1Qnty() {return meal1Qnty; }
	public void setMeal1Qnty(int meal1Qnty) {
		this.meal1Qnty = meal1Qnty; setTotPrice(); setTotToPay();}
	public int getMeal2Qnty() { return meal2Qnty; }
	public void setMeal2Qnty(int meal2Qnty) {
		this.meal2Qnty = meal2Qnty; setTotPrice(); setTotToPay();}
	public int getMeal3Qnty() { return meal3Qnty; }
	public void setMeal3Qnty(int meal3Qnty) {
		this.meal3Qnty = meal3Qnty; setTotPrice(); setTotToPay();}
	public int getMeal4Qnty() { return meal4Qnty; }
	public void setMeal4Qnty(int meal4Qnty) {
		this.meal4Qnty = meal4Qnty; setTotPrice(); setTotToPay();}
	public String getVoucherCode() { return voucherCode; }
	public void setVoucherCode(String voucherCode) {
		this.voucherCode = voucherCode;
	}
	public int getNumber() { return number; }
	public float getTotPrice() { return totPrice; }
	public float getDiscount() { return discount; }
	public float getTotToPay() { return totToPay; }
}
```