Arrays, Strings, Basic Java:

1. Write a Java program that takes 5 string inputs then outputs them **alphabetically sorted**. You are allowed to use compareTo() method.

Sample Input	Sample Output
ABC **AAC** **FLL** **CDE** **AVB**	**AAC, ABC, AVB, CDE, FLL**

ABC **XYZ** **CBA** **AAA** **ZZY**	**AAA, ABC, CBA, XYZ, ZZY**

2. Write a Java program which takes one floating number (three decimal precision) and prints the integer and decimal part separately.

Sample Case 1
Sample input: **10.546**
Sample Output: **10** **546**

Sample Case 2
Sample input: **-101.875**
Sample Output: **-101** **875**

3. Write a Java program that takes final marks of 5 students as input and groups them by their grade. Do not print blank/empty groups.

80	-	100	=	A
65	-	<80	=	B
50	-	<65	=	C
		<50	=	F

Sample Input	Sample Output
65 **87** **44** **97** **79**	**A: 87, 97** **B: 65, 79** **F: 44**

66 **58** **75** **0** **17**	**B: 75** **C: 66, 58** **F: 0, 17**

OOP Codes:

4. Write the code for **Account** class that generates the following output:
Note: Assume that each account must have at least 100 tk remaining. If a withdrawal amount results in the account balance getting under 100 tk, withdrawal will be unsuccessful.

Driver	**Output**

public class test1 {
  public static void main(String[] args) {
  Account p1 = new Account();
  p1.setUp("Abdul", "Service Holder", 500000);
  p1.addMoney(300000);
  p1.printDetails();
  System.out.println("========================");
  Account p2 = new Account();
  p2.setUp("Rahim", 700000);
  p2.withdrawMoney(700000);
  p2.printDetails();
  System.out.println("========================");
  Account p3 = new Account();
  p3.setUp("Ashraf", "Govt. Officer", 200000);
  p3.withdrawMoney(250000);
  p3.printDetails();
  System.out.println("========================");
  }
}
	Account for Abdul created successfully.
	Name: Abdul
	Occupation: Service Holder
	Balance: 800000
	========================
	Account for Rahim created successfully.
	Withdraw Unsuccessful.
	Name: Rahim
	Occupation: Self-employed
	Balance: 700000
	========================
	Account for Ashraf created successfully.
	Withdraw Unsuccessful.
	Name: Ashraf
	Occupation: Govt. Officer
	Balance: 200000
	========================

5. Write the code for **Smartphone** class:

Driver	**Output**

public class test2 {
  public static void main(String[] args) {
  Smartphone s1 = new Smartphone();
  s1.printDetail();
  s1.features = new String[5];
  System.out.println("========================");
  s1.addFeature("Display", "6.1 inch");
  System.out.println("========================");
  s1.setModel("Samsung Note 20");
  s1.addFeature("Display", "6.1 inch");
  s1.printDetail();
  System.out.println("========================");
  Smartphone s2 = new Smartphone();
  s2.printDetail();
  s2.features = new String[5];
  s2.setModel("Iphone 12 Pro");
  s2.addFeature("Display", "6.2 inch");
  s2.addFeature("Ram", "6 GB");
  System.out.println("========================");
  s2.printDetail();
  s2.addFeature("Display", "Amoled panel");
  s2.addFeature("AirDrop");
  System.out.println("========================");
  s2.printDetail();
  System.out.println("========================");
  }
}
	Phone Model: null
	==============================
	Feature can not be added without model name
	==============================
	Phone Model: Samsung Note 20
	- Display: 6.1 inch
	==============================
	Phone Model: null
	==============================
	Phone Model: Iphone 12 Pro
	- Display: 6.2 inch
	- Ram: 6 GB
	==============================
	Phone Model: Iphone 12 Pro
	- Display: 6.2 inch
	- Ram: 6 GB
	- Display: Amoled panel
	- AirDrop
	==============================

6. Write the code for **Contact** class. Here, each contact object saves call and sms history from both from and to contact numbers.
(Recommended to try after editing the history size limit)

Driver	Output

public class test10 {
  public static void main(String[] args) {
  Contact c1 = new Contact();
  c1.number = "+880-1111";
  c1.callHistory = new String[3];
  c1.smsHistory = new String[2];
 	
  Contact c2 = new Contact();
  c2.number = "+880-2222";
  c2.callHistory = new String[2];
  c2.smsHistory = new String[3];
 	
  c1.sendSms(c2, "Good morning");
  c2.call(c1);
  System.out.println("-- -- -- -- --");
  c2.sendSms(c1, "Call when you're free");
  c1.call(c2);
  System.out.println("-- -- -- -- --");
  c1.showHistory();
  System.out.println("-- -- -- -- --");
  c2.showHistory();
  System.out.println("-- -- -- -- --");
 	
  c2.sendSms(c1, "Hello!");
  c1.call(c2);
  System.out.println("-- -- -- -- --");
  c1.showHistory();
  c2.showHistory();
  c1.sendSms(c1, "Are you free?");
  c2.call(c2);
  System.out.println("-- -- -- -- --");
  }
}
	SMS sent from +880-1111 to +880-2222
	+880-2222 called +880-1111
	-- -- -- -- --
	SMS sent from +880-2222 to +880-1111
	+880-1111 called +880-2222
	-- -- -- -- --
	Call History for +880-1111:
	+880-2222
	+880-2222
	SMS History for +880-1111:
	from +880-1111 to +880-2222: Good morning
	from +880-2222 to +880-1111: Call when you're free
	-- -- -- -- --
	Call History for +880-2222:
	+880-1111
	+880-1111
	SMS History for +880-2222:
	from +880-1111 to +880-2222: Good morning
	from +880-2222 to +880-1111: Call when you're free
	-- -- -- -- --
	Maximum messages reached for +880-1111, please recharge more.
	Maximum calls reached for +880-2222, please recharge more.
	-- -- -- -- --
	Call History for +880-1111:
	+880-2222
	+880-2222
	SMS History for +880-1111:
	from +880-1111 to +880-2222: Good morning
	from +880-2222 to +880-1111: Call when you're free
	-- -- -- -- --
	Call History for +880-2222:
	+880-1111
	+880-1111
	SMS History for +880-2222:
	from +880-1111 to +880-2222: Good morning
	from +880-2222 to +880-1111: Call when you're free
	Maximum messages reached for +880-1111, please recharge more.
	Maximum calls reached for +880-2222, please recharge more.
	-- -- -- -- --

7. Carefully read the following Driver/Tester code and corresponding output to identify the attributes and methods for the **Club** class. Afterwards, Design the **Club** class:
   Assume that at max, 3 events are possible.

Driver	**Output**

public class ClubTester {
  public static void main(String[] args) {
  Club club1 = new Club();
  System.out.println("1================");
  System.out.println(club1.approveClub("Makers Club",4,10000));
  System.out.println("2================");
  System.out.println(club1.approveClub("Makers Club",10,10000));
  System.out.println("3================");
  club1.info();
 	
  System.out.println("4================");
  club1.createEvent("Exhibit", 4099, 5);
  System.out.println("5================");
  club1.createEvent("Impromptu", 5700, 6);
  System.out.println("6================");
  club1.recruitMember(5);
  System.out.println("7================");
  club1.createEvent("Impromptu", 5700, 6);
  System.out.println("8================");
  club1.info();
  System.out.println("9================");
  club1.createEvent("Potluck", 1200, 3);
  System.out.println("10================");
  club1.createEvent("Potluck", 100, 3);
  System.out.println("11================");
  club1.info();
  System.out.println("12================");
  club1.createEvent("Speech", 100, 2);
  System.out.println("13================");
  club1.endEvent("Exhibit");
  System.out.println("14================");
  club1.info();
  System.out.println("15================");
  club1.createEvent("Speech",100, 2);
  }
}
	1================
	A club must have at least 5 members
	2================
	New club, Makers Club, created with 10 members.
	3================
	Name of club: Makers Club
	Non-working members: 10
	Current Budget: 10000.0
	No events yet.
	4================
	New event, "Exhibit" has started!
	5 out of 10 available members are now working.
	5================
	Need 1 more member(s) to arrange.
	6================
	New members recruited
	Total non-working members now are 10.
	7================
	New event, "Impromptu" has started!
	6 out of 10 available members are now working.
	8================
	Name of club: Makers Club
	Non-working members: 4
	Current Budget: 201.0
	2 Events:
	Exhibit:5, Impromptu:6
	9================
	Not enough budget.
	10================
	New event, "Potluck" has started!
	3 members out of 4 are now working.
	11================
	Name of club: Makers Club
	Non-working members: 1
	Current Budget: 101.0
	3 Events:
	Exhibit:5, Impromptu:6, Potluck:3
	12================
	Need 1 more member(s) to arrange.
	13================
	Exhibit has ended!
	5 members are free now.
	14================
	Name of club: Makers Club
	Non-working members: 6
	Current Budget: 101.0
	2 Events:
	Impromptu:6 Potluck:3
	15================
	New event, "Speech" has started!
	2 members out of 6 are now working.

OOP Tracing:

8. Find the output after running the following code:

```
1  public class tracing1 {
2      public static void main(String[] args) {
3          Test m = new Test();
4          m.n = m.m = 5;
5          Test n = new Test();
6          n.m = m.metA(2);
7          n.n = n.metA(4);
8          System.out.println(m.n+m.m+" "+n.m+" "+n.n);
9      }
10 }
11
12 class Test {
13     int m, n = 1;
14
15     int metA(int n){
16         n += m + 3;
17         int s = n+ this.n;
18         if (s%2 == 0) return s;
19         Test t = new Test();
20         t.n = (++this.m) - (++m) + t.m;
21         this.n = n + t.metA(t.m);
22         System.out.printf("%d %d %d\n", m, n, s);
23         return s+this.n;
24     }
25 }
```

Outputs		