Problem 1

Write a method called `mostFrequentChar` that takes one parameter, a `String s`, and does the following:

1. Returns the character that occurs the most number of times in the `String s`. If there is a "tie" for the most frequently occurring character, return the character that occurs the earliest in the `String`.
2. You may use **at most** one loop in your written solution.
3. See sample usage below.

Sample usage:

```
String s1 = "banana";

mostFrequentChar(s1) would return:
a
```

```
String s2 = "g8ueixbo927v";

mostFrequentChar(s2) would return:
g
```

```
String s3 = "brookkeeps";

mostFrequentChar(s3) would return:
o
```

Problem 2
Write the method `notThere` that takes two parameters, a `String s`, and a second String `s2`. Your method is to return an integer representing the number of characters in String `s2` that do not appear in String `s`.

Note: String `s` and String `s2` may have duplicates.

You may use **at most** one loop in your written solution.

<u>Problem 3</u>
Write the following method:

```
public static String eliminate(String str, char c1, char c2, char c3)
```

The method method receives a string and three single characters. The method is to return a new string that is made up of all characters of the original string except those characters stored in the variables c1, c2 and c3.

You may use at most one loop in your written solution.

Sample data:

str	c1	c2	c3	method returns
“abcdefgh”	a	d	g	bcefh
“mary mary quite”	y	‘ ‘ (a space)	q	marmaruite

Problem 4

Write a Java program named Exceptions.

The program is to prompt the user to enter an integer, and is to do the following:

1) if the user input value is equal to 0, the program is to throw a NullPointerException and display the message “reference variable must point at an object;
2) if the user input value is less than 0, the program is to throw an IllegalStateException and display the message “cannot use a negative value”
3) if the user input value is greater than 10, the program is to throw an IllegalArgumentException and display the message “cannot use zero”;
4) if the user input value is 999, the program is to throw an ArrayIndexOutOfBoundsException and display the message “must stay within the array bounds”;
5) if the user input value is 3000, the program is to throw an UnsupportedOperationException and display the message “hmm, not yet implemented”.

Otherwise, the progam will display the value entered by the user.

Extra Credit: how would the FileNotFoundException, which is to display the message “wrong file name” be handled.

Problem 5

Please write the exact output of class PlaneApp.

```java
public class Plane
{
   private String name;
   private int callNumber;

   public Plane(String n)
   {
      this(n, 000);
   }

   public Plane(String n, int identifier){
      name = n;
      callNumber = identifier;
   }
   public String getName(){
      return name;
   }
   public int getCallNumber(){
      return callNumber;
   }
   public void setCallNumber(int cn){
      callNumber = cn;
   }
   public String toString(){
      displayProperties();
      return "Plane: " + name;
   }
   public void displayProperties(){
      System.out.println("One Classy Plane");
   }
}
```

```java
public class CargoPlane extends Plane
{
   private String type ; //names: commercial, private

   public CargoPlane(){
      super("Bravo", 239); //name, call number
      type="commercial";
   }
   public CargoPlane(String n, String t){
      super(n, 727);
      type = t;
   }
   public String getType(){return(type);}

   public void displayProperties(){
      System.out.println(super.getCallNumber());
      System.out.println("Very Fast");
      super.displayProperties();
   }

   public boolean equals(Object o){
      System.out.println("Fancy Plane");
      CargoPlane cpThis = (CargoPlane)this;
      CargoPlane cpThat = (CargoPlane)o;

      if(!(o.getClass().equals(this.getClass())))
         System.out.println("crashing");
      else
         System.out.println("phew, not crashing");

      return(cpThis.getName().equals(cpThat.getName()));
   }
}
```

```java
public class PlaneApp
{
   public static void main(String [] args){
      Plane p = null;
      Plane cc = new Plane("Coolness Cruiser");
      System.out.print("Plane p is: " );
      System.out.println(p);
      System.out.print("Plane cc is: ");
      System.out.println(cc);
      System.out.println("/**************************************/");
      CargoPlane cp1 = new CargoPlane();
      CargoPlane cp2 = new CargoPlane("Dottie", "Commercial");
      Plane p1 = new CargoPlane("Chug", cp1.getType());
      Plane p2 = new Plane("Echo", 787);
      Plane [] planes = {p, cp1, cp2, p1, p2, null};
      System.out.println("/**************************************/");
      planes[0] = cp2;
      System.out.println(cp1.equals(planes[0]));
      cp1.displayProperties();
      System.out.println(planes[0]);
      System.out.println("Fly the friendly skies");
      System.out.println("/**************************************/");
   }
}
```

Output Window

Problem 6

Given the below interface and abstract class, create the non-abstract class `TicTacToe` as follows:

1. `TicTacToe` class inherits from the `Game` class and implements the `Board` interface.
2. The `TicTacToe` constructor should take one `String` parameter and one `int` parameter and set the superclass instance variables.
3. `TicTacToe` should have a properly encapsulated 2D `char` array instance variable named `board`.
4. The `create` method should create a 2D `char` array of size `r` rows and `c` columns, composed of all hyphens ('`-`'), and set the `board` instance variable to that array.
5. The `play` method takes an `int` parameter representing the player (player 1, player 2, etc). The method should prompt the user to enter the row and the column number of where they want to play on the board. You can assume that the `Scanner` object has been created (see instructions on first page). Modify the `board` instance variable to contain an '`X`' or an '`O`' at the specified row and column. Player 1 is '`X`' and player 2 is '`O`'.
6. Override the `printGameInfo` method to print the Game name on the first line, the number of players on the second line, and then the `board` values on the following three lines.
7. **You should write the code to produce the output in the same format provided below. Do not add any additional methods other than the ones described in the instructions.**
8. Complete and place your code in the box provided on the next page. **Any code on this page will not be considered as part of your solution.**

Game.java

```java
public abstract class Game
{
  private String name;
  private int players;

  protected Game(String n, int p)
  {
    this.name = n;
    this.players = p;
  }

  public abstract void play(int p);

  public void printGameInfo()
  {
    String s = this.name + " Players: "
                 + this.players;
    System.out.println(s);
  }
}
```

Board.java

```java
public interface Board
{
  public abstract void create(int r, int c);
}
```

Sample Usage

```java
TicTacToe t = new TicTacToe("TicTacToe", 2);
t.create(3, 3);
t.printGameInfo();
t.play(1);
t.play(2);
t.printGameInfo();
System.out.println();
t.play(1);
t.play(2);
t.printGameInfo();
```

Sample Output

```
TicTacToe Players: 2
- - -
- - -
- - -
Enter row: 1
Enter column: 1
Enter row: 0
Enter column: 2
TicTacToe Players: 2
- - O
- X -
- - -

Enter row: 0
Enter column: 0
Enter row: 2
Enter column: 2
TicTacToe
X - O
- X -
- - O
```

STUDENT ID ____________________

Problem 7

Complete the `divisibleby10()` method below so that it does the following:

1. Reads the input from the `numbers.txt` file (sample provided below). You do **not** know how many lines are in the file. The sample provided is just an example of the file format.
2. Determines whether the number on each line is evenly divisible by `10`. If it is, print out the result of dividing the number by `10`. Otherwise, print out "`Not divisible by 10`". Be careful when working with large integers!
3. Keep a count of the numbers that are divisible by `10` and display that value (matching the sample output format exactly) after processing the entire file. Note that the sample output displays this value as "`xxx`" since only a sample of the file has been provided.
4. Handles a `FileNotFoundException` by printing out "`File not found`".
5. You can assume that the `Scanner` and `File` classes have been imported.
6. Your solution must be able to handle integers of any size.
7. Sample usage is provided below. Your output format should match the sample output format **exactly**.
8. Complete and place your code in the box provided on the next page. **Any code on this page will not be considered as part of your solution.**

Sample of file:

```
2300000000
125
5791029887593372947833927311138273
250
...
```

Sample output:

```
230000000
Not divisible by 10
Not divisible by 10
25
...
Count: xxx
```