MULTIPLE CHOICE QUESTIONS 1-10. (4 points each)

1. Which of the following is true with respect to Vehicle and Car?

```
public class Vehicle {
   protected String name;
   private int ID;
}
public class Car extends Vehicle {
   protected int miles;
}
```

   a. Car members have access to Vehicle ID
   b. Vehicle members have access to Car miles
   c. Car members do not have access to any Vehicle members
   d. Car members have access to Vehicle name
   e. None of the above

2. Which lines will not compile?

```
public class Vehicle {
   protected String name;
   private int ID;
   public void setID(int pID) {...}
}
public class Car extends Vehicle {
   private int miles;
   public Car() {
1     miles = 145;
2     setID(99);
3     ID = 47;
4     name = "Honda";
   }
}
```

   a. lines 1 and 3
   b. there are no errors
   c. line 3 only
   d. lines 3 and 4
   e. None of the above

3. A derived class method inside a child class with the same name, parameters, and return type as a base class method is said to_____the base class's method.

   a. overload
   b. override
   c. inherit
   d. copy
   e. None of the above

4. Which is always true about java classes?

   a. A class can implement only one interface
   b. A class can extend from multiple classes
   c. An interface can implement multiple classes
   d. A class can implement multiple interfaces
   e. None of the above

5. Given the following code, what is the output?

```
public class TestShape {
   public void what() { System.out.print("Shape ");}

   public static void main(String[] args) {

      Shape[] shapes = {new Shape(), new Rectangle(), new Square(),new Circle()};
      for (Shape s : shapes){
         s.what();
         System.out.print(" ");
      }
   }

}
class Shape {
   public void what() { System.out.print("Shape ");}
}
class Rectangle extends Shape {
   public void what() { System.out.print("Rectangle "); }
}

class Square extends Rectangle {
}

class Oval extends Shape {
   public void what() { System.out.print("Oval "); }
}

class Circle extends Oval {
   public void what() { System.out.print("Circle ");}
}
```

a. Shape Shape Shape Shape
b. Shape Rectangle Square Circle
c. There will be a compile time error
d. Shape Rectangle Rectangle Circle
e. None of the above

6. Given the following program:

```
import java.util.*;
public class Main{
  public static void main(String[] args){
    try{
         int m = Integer.parseInt(args[1]);
         int n = Integer.parseInt(args[2]);
         System.out.println(m + "/" + n + " = " + (m/n));
         System.out.println("The division is done");
    }
    catch(ArithmeticException e){
       System.out.println("Division by zero");
    }
    catch(InputMismatchException e){
       System.out.println("Strings detected");
```

```
    }
    System.out.println("End of the program.");
  }
}
```

What is the output if you run the program using:
`java Main 1 0 1 1`

a. Strings detected
   End of the program
b. 0/1 = 0
   The division is done
   End of the program.
c. Division by zero
   End of the program.
d. Division by zero
   Strings detected
   End of the program.
e. None of the above

7. Show the output of the following code:

```
public class Test7 {

  public static void main(String[] args) {
      int[] first = {1, 0, -1};
      int[] second = {1, 2, 1};

      bigSwap(second,first[1]);
      System.out.println(first[1] + " " + second[1]);
      }

  public static void bigSwap(int[] first, int second) {
       int temp=first[1];
       first[1]=second*2;
       second=temp;
       }
}
```

a. 1 2
b. 1 1
c. 0 2
d. 2 1
e. None of the above

8. If you instantiate an abstract class, the class or object you wind up with

a. is also an abstract class
b. you can't instantiate an abstract class
c. is a reference to an object
d. is an interface
e. None of the above

9. Given the classes below, what is the output of the following program execution:

```
public class BadCharException extends Exception
{
 public BadCharException (String message)
 {
  super (message);
 }
}
```

```
public class TestNames
{
public static void main (String[] args)
{
  String[] names={"one", "two", "three"};
  BadCharException problem=new BadCharException("!");

  try
  {
    for (String name: names){
      if (name.charAt(0) !='o') throw problem;
      System.out.print(name.charAt(0));

      }
   }
   catch (BadCharException e){ System.out.print(e.getMessage());
   }
   catch (Exception e){ System.out.print("?");
   }
  }
}
```

a. `o!?`
b. `o!`
c. `?!!`
d. `o!!`
e. None of the above

10. If the method is invoked as `recursive(5,2)`, what is returned?

```
public static int recursive(int a, int b)
   {
      if (a == b)
         return 0;
      else
         return recursive(a-1, b) + b;
   }
```

a. This is an infinite recursion
b. 6
c. 9
d. 4
e. None of the above

11. **(20 points)** Given the `CoffeeDrink` class defined by two Ingredient objects contained in it (coffee and milk) as well as the Ingredient class representing an ingredient in terms of its name and its strength. Strength is an integer between 1-5, 5 being the highest strength. The lower the ratio of milk strength to coffee strength, the stronger the coffee drink.

```
public class Ingredient{
   private String name;
   private int strength;

   public Ingredient(String name, int strength)
   {
      this.name=name; this.strength=strength;
   }

   public int getStrength(){return strength;}

   public String toString(){return name+" strength: "+strength;}
}

public class CoffeeDrink
{
   private Ingredient coffee, milk;

   public CoffeeDrink(Ingredient i1, Ingredient i2)
   {
      this.coffee=i1;
      this.milk=i2;
   }

   public Ingredient getIngred1(){return coffee;}

   public Ingredient getIngred2(){return milk;}

   // compares two drinks based on their milk/coffee ratio.
   //drink 1 is stronger than drink 2 if that ratio is smaller than
   // the corresponding ratio of drink 2
   public int compareTo(CoffeeDrink other)
   {
      if ((double)milk.getStrength()/coffee.getStrength() > (double)
other.getIngred2().getStrength()/other.getIngred1().getStrength())
         return 1;
      else if ((double)milk.getStrength()/coffee.getStrength() < (double)
other.getIngred2().getStrength()/other.getIngred1().getStrength())
         return -1;
      else
         return 0;
   }

  public String toString(){ return coffee.toString()+"\n"+milk.toString();}
}
```

Write a method `testCoffeeDrinks` that takes one parameter:
- `drinks`, an non-empty array of CoffeeDrink objects

The method finds and prints to the screen the strongest drink (You must use the compareTo method for full credit). If more than one such drink occur in the array, pick the first one. The method also creates, populates and returns an array of integers containing the frequency of each possible coffee strength (1-5), in the drinks array. For example, if the information of the drinks in the array is given in the table below:

Coffee Drink Number	Ingredient1	Ingredient2
0	coffee, 4	milk, 3
1	coffee, 2	milk, 3
2	coffee, 2	milk, 1
3	coffee, 4	milk, 1
4	coffee, 3	milk, 2
5	coffee, 4	milk, 4

The program will display:
```
coffee strength:4
milk strength: 1
```

And return the array `{0,2,1,3,0}`

12. **(20 points)** Write a **recursive** method called **`checkString`** that checks whether a string contains same characters. The method takes 1 parameter, `s`, a string (assume any letters are all lower case). The method returns true if all characters in the string are the same and false otherwise.

For example,

```
…
   System.out.println(checkString("eel"));  //will print false
   System.out.println(checkString("eeee")); //will print true
   System.out.println(checkString("11"));    //will print true
   System.out.println(checkString("cs113"));//will print false
…
```

13. **(20 points)** The abstract class `CollegeApplication` below defines a particular college application of a student in terms of the `stdName` (a String) and the `major` (a String), the applicant is declaring.

    Write the definition of a child class of class `CollegeApplication` named `UGCollegeApplication` representing an application of the undergraduate school. A student application for any major within the college is accepted if he/she has an SAT score of at least 1285 and a GPA no less than 3.0. The `UGCollegeApplication` defines the following members:

    - two attributes, `satScore` (an integer) as well as `gpa` (a double), representing the student's record
    - a non-default constructor for the class that initializes all attributes.
    - a getter for the `satScore`
    - a setter for the `gpa`
    - a method `betterAcceptChanceThan()` that compares two `UGCollegeApplication` objects and returns `true` if the current major application has the highest `satScore` and `gpa` values than the other and returns `false`, otherwise.

HINT: Remember to define/redefine any other methods needed in the child class.

```
public abstract class CollegeApplication{

  protected String stdName;
  protected String major;

  public CollegeApplication(String name, String major){
    this.stdName=name;
    this.major=major;
  }

  public String getStudentName(){ return stdName;}
  public String getMajor() { return major; }

//  public void setDestination(String name){ satScore=name;}
//  public void setBudget(int gpa){this.gpa=gpa;}

  public abstract boolean isAccepted();
 //returns true/false depending on whether the student's application is accepted

  public String toString(){   //returns the details of the current application
     return stdName+" major: "+major+ " - no addmission status";
 }
}
```

Quick Reference

Scanner Class

Scanner(InputStream source)
Scanner(File source)
Scanner(String source)
String next()
String nextLine()
int nextInt()
double nextDouble()
float nextFloat()

String Class

String(String str)
int length()
int compareTo(String anotherString)
char charAt(int index)
boolean equals(String anotherString)
String substring(int beginIndex, int endIndex)
String substring(int beginIndex)

Random Class

Random()
float nextFloat()
int nextInt(int num)
int nextInt()

Math Class

static double random()
static final double PI