**Overloading** :

Method Overloading is a feature that allows a class to have two or more methods having same name, if their argument lists are different.

**Argument lists could differ in –**  
1. Number of parameters.  
2. Data type of parameters.  
3. Sequence of Data type of parameters.

**Question No. 1**

Write a class named as “AddNumberClass”.

1. Define method named as “addNumbers” with two parameters first of type **int** and second of type **float** and method return type **float**.That method will calculate sum of two numbers and return total sum in **float** type.
2. Define another method named as “addNumbers” with two parameters first of type **float** and second of type “**int**” and method return type **float.**That method will also calculate sum of two numbers and return total sum in **float** type.
3. Define another method named as “addNumbers” with two parameters first of type **int** and second of type “**int**” and method return type **int.**That method will also calculate sum of two numbers and return total sum in **int** type.
4. Call above methods one by one in main() using class **AddNumberClass** object.
5. Inside main() write as :

AddNumberClass addNumber = new AddNumberClass();

addNumber. addNumbers(5,5); ---🡪 which method will be called here ? a , b or c

addNumber. addNumbers(5,5.0);----🡪 which method will be called here ? a , b or c

addNumber. addNumbers(5.0,5);---🡪 which method will be called here ? a , b or c

**Question No 2**

Overloading Quiz:

Lets see few Valid/invalid cases of method overloading:

Solve it and give reasons in a word file after each question and push your file to your github repository:

**Case 1:**

**int mymethod(int a, int b, float c)**

**int mymethod(int var1, int var2, float var3)**

**Question: Are the above two examples of Overloading ? Will it result any error?**

**They both are examples of overloading. But problem is that they won’t run together in one class. it will result in error because both are using same types. Error: Both are already defined (int, int, float). Also you will have to provide a return statement.**

**Case 2:**

**int mymethod(int a, int b)**

**int mymethod(float var1, float var2)**

**Question: Are the above two examples of Overloading ? Explain with reasons.**

**Yes they are example of method overloading. We are overloading two ints and floats. Also you will have to provide a return statement.**

**Case 3:**

**int mymethod(int a, int b)**

**float mymethod(int var1, int var2)**

**Yes they are examples of overloading. Also you will have to provide a return statement.**

**Question: Are the above two examples of Overloading ?Will it give any error?**

Inheritance:

**Question No. 3**

Create a class named as “Person” with common properties myName, myAge, myGender.Create its getters and setters methods.

Write a Teacher class that extends the parent class Person.

1. Add instance variables to the class for *subject* (e.g. “Computer Science”, "Chemistry", "English", "Other”) and *salary* (the teacher’s annual salary). *Subject* should be of type String and salary of type **double**. Choose appropriate names for the instance variables.
2. Write a constructor for the Teacher class. The constructor will use five parameters to initialize myName, myAge, myGender, *subject*, and *salary*. Use the **super** reference to use the constructor in the Person superclass to initialize the inherited values.
3. Write “setter” and “getter” methods for all of the class variables. For the Teacher class they would be: getSubject, getSalary, setSubject, and setSalary.
4. Write the toString() method for the Teacher class. Use a **super** reference to do the things already done by the superclass.

**Question No. 4**

Write a CollegeStudent subclass that extends the Student class.

* 1. Add instance variables to the class for major (e.g. “Electrical Engineering”, “Communications”, “Undeclared”) and year (e.g. FROSH = 1, SOPH = 2, ...). *Major* should be of type String and*year* of type **int**. Choose appropriate names for the instance variables.
  2. Write a constructor for the CollegeStudent class. The constructor will use seven parameters to initialize myName, myAge, myGender, myIdNum, myGPA, *year*, and *major*. Use the**super** reference to use the constructor in the Student superclass to initialize the inherited values.
  3. Write “setter” and “getter” methods for all of the class variables. For the CollegeStudent class they would be: getYear, getMajor, setYear, and setMajor.
  4. Write the toString() method for the CollegeStudent class. Use a **super** reference to do the things already done by the superclass.

**Question No 5**

Write a testing class with a main() that constructs all of the classes (Person, Student, Teacher, and CollegeStudent) and calls their toString() method. Sample usage would be:

Person bob = **new** Person("Coach Bob", 27, "M");  
System.out.println(bob);

Student lynne = **new** Student("Lynne Brooke", 16, "F", "HS95129", 3.5);  
System.out.println(lynne);

Teacher mrJava = **new** Teacher("Duke Java", 34, "M", "Computer Science", 50000);  
System.out.println(mrJava);

CollegeStudent ima = **new** CollegeStudent("Ima Frosh", 18, "F", "UCB123", 4.0, 1, "English");  
System.out.println(ima);