King Saud University

College of Computer and Information Sciences

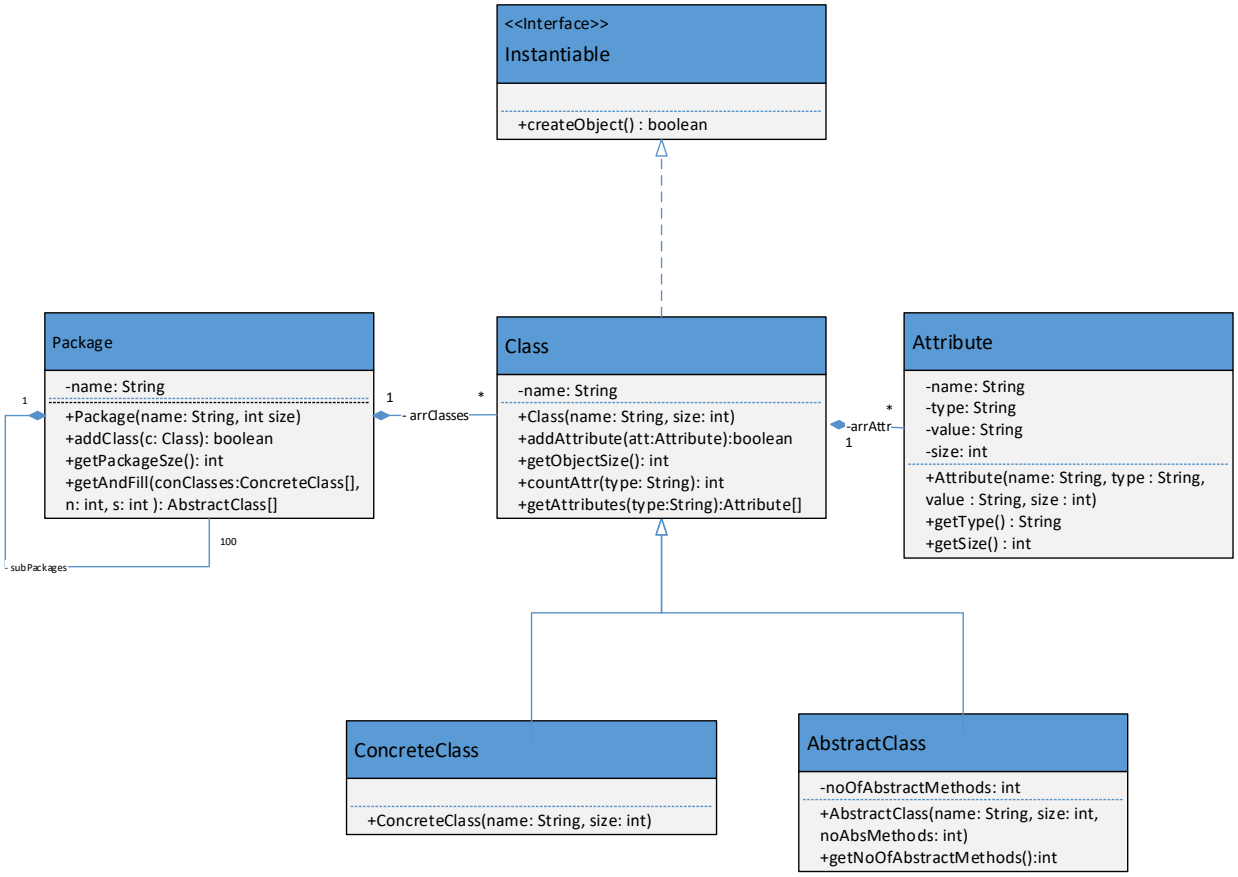
Computer Science Department

Course Code:	CSC 113
Course Title:	Computer Programming II
Semester:	Spring 2019
Exercises Cover Sheet:	**Midterm 2 Exam**

Student Name:	
Student ID:	
Student Section No.	

Tick the Relevant	Computer Science B.Sc. Program ABET Student Outcomes	Question No. Relevant Is Hyperlinked	Covering %
X	**a) Apply knowledge of computing and mathematics appropriate to the computer science;**		
	b) Analyze a problem, and identify and define the computing requirements appropriate to its solution		
X	**c) Design, implement and evaluate a computer-based system, process, component, or program to meet desired needs;**		
X	**d) Function effectively on teams to accomplish a common goal;**		
	e) Understanding of professional, ethical, legal, security, and social issues and responsibilities;		
	f) Communicate effectively with a range of audiences;		
	g) Analyze the local and global impact of computing on individuals, organizations and society;		
	h) Recognition of the need for, and an ability to engage in, continuing professional development;		
X	**i) Use current techniques, skills, and tools necessary for computing practices.**		
	j) Apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices;		
	k) Apply design and development principles in the construction of software systems of varying complexity;		

Exercise 1:



The interface Instantiable:

- Methods:
  - ***createObject()*:**
    - For ***ConcreteClass***: it will return true.
    - For ***AbstractClass***: it will return false.

The class Attrbute:

- Attributes:
  - ***name:*** the name of the attribute.
  - ***type***: the type of the attribute.
  - ***value***: the value of the attribute.
  - ***Size***: size of the attribute in bytes
- Methods:
  - ***Attribute (…)***: constructor.
  - ***getType()*:** this method returns the type of the attribute.
  - ***getSize()***: This method will return the size of the attribute.

The class Class:

- Attributes:
  - ***name:*** the name of the class.
- Methods:
  - ***Class (…)***: constructor.
  - ***addAttribute(att:Attribute***): This method adds attribute in the class. On successful addition it will return true, otherwise false.
  - ***getObjectSize()*:** This method returns the size of the object by adding all the attribute sizes.
  - ***countAttr(type:String)***: This method will return the number of attributes with the type ***type***.
  - ***getAttributes(type:String)***: This method will return an array of attributes with type ***type***. The length of the returned array should not be more than the number of attributes in the object of the type ***type***.

The class ConcreteClass:

- Methods:
  - ***ConcreteClass (…):*** constructor.

The class AbstractClass:

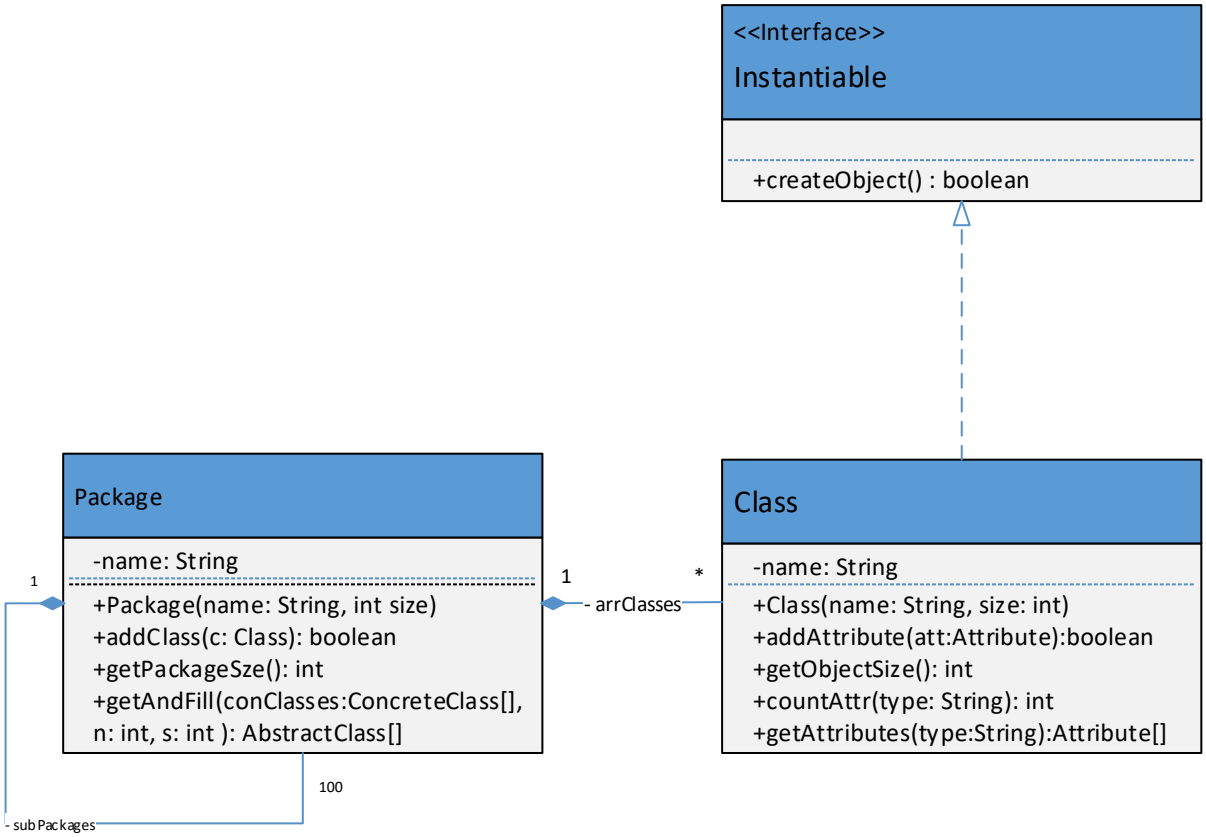
- Attributes:
  - ***noOfAbstractMethods***: number of abstract methods in the class.
- Methods:
  - ***AbstractClass (…):*** constructor.
  - ***getNoOfAbstractMethods():***this method returns number of abstract methods in the class.

QUESTION: Translate into Java code:

1. The interface ***Instantiable***,
2. The class ***Attribute***
3. The class ***Class.***
4. The class ***AbstractClass.***

Exercise 2:

Let's consider the class ***Class*** and its subclasses as described in exercise 1.



The class ***Package:***

- Attributes:
  - ***name***: the name of the Package.
- Methods:
  - ***Package(…)***: constructor.
  - ***addClass(c: Class):*** This method adds the Class ***c*** to the current package. This method returns true if the Class c is successfully added. Otherwise, it returns false.
  - ***getPackageSize():*** This recursive method returns the total size of the Package. The total size is computed as following:

    $$\textit{Total size of Package} = \sum \textit{size of objects} + \sum \textit{size of subpackages}$$

  - ***getAndFill***(***conClasses:*** *ConcreteClass[]*, ***n***: int, ***s***:int): This method puts the ConcreteClass objects having an object size greater than ***s*** in the conClass array, and it will return an array of the objects of AbstractClass that have number of abstract methods equal to ***n***.

QUESTION: Translate into Java code the class ***Package.***