KAZAKH KBTU BRITISH TECHNICAL UNIVERSITY

Object-Oriented programming

Laboratory work #3
Inheritance, polymorphism, abstract classes

Problem #1

Create class `Animal` and a derived class of Animal at your choice (Cat, dog, crocodile, etc…). In a subclass (or derived class) demonstrate :

- the methods `overriding and overloading` of the base class methods.
- The use of `super()` keyword with and without parameters.

Problem #2

Create the abstract class for 3D shapes, e.g., `volume`(), `surfaceArea`() (add other methods at your choice!). Then create data types `Cylinder, Sphere, Cube` extending this class.

Problem #3

Create a class called `Employee` whose objects are records for an employee. This class will be a derived class of the class `Person` (MUST contain `equals` and `toString` methods).

An employee record has an employee's name (inherited from the class Person), an annual `salary` represented as a single value of type double, a `year` the employee started work as a single value of type int and a national `insuranceNumber`, which is a value of type String. Inside this class you need to override `toString` and `equals` methods of the `Person` class.

Your class should have a reasonable number of constructors and accessor methods. Then create a class `Manager` extending `Employee`, each manager has a team of Employees (Vector) and can get a bonus. You need to override `toString` and `equals` methods. Write another class containing a main method to fully test your class definition.

*Use super() keyword whenever possible. Otherwise you will lose points.

*Check the quality of your equals and hashcode methods by adding several employees to a `HashSet` and checking whether it allows duplicate items.

Create a data type for chess pieces. Inherit from the base abstract class `Piece` and create subclasses `Rock`, `King` and so on. Include a method `isLegalMove(Position a, Position b)` that determines whether the given piece can move from a to b.

Bonus (1 point, but can be more):

Make a class `Board` and some test class in order to fully imitate chess game. Think of how you will store the current state of the game, take moves from user, drawing the board on a console, checking for illegal moves, etc.

Problem # 5

Create classes according to the following diagram.

*In your test program, you have to offer a user to either print info about all added persons or add a new person (who can be either a `Person`, `Student`, or `Employee`).

*Store all users in a `HashSet` or `Vector`.