Problem 1

Write a class Square representing squares and containing one private field side and

- constructor initializing the field side;
- method `public double getSide()` returning the length of the side of the square;
- method `public double getArea()` returning the area of the square;
- method `public double getDiagonal()` returning the length of the diagonal of the square.
- method `public double getPerimeter()` returning the length of the perimeter of the square.

In the **main** function *of another class* **Main** create a few objects of class **Square** and test all functions.

Problem 2

Create two classes, **Square** (with a field corresponding to the length of the side) and **Circle** (with the field describing the radius). Both classes should have one-parameter constructors, methods returning side/radius (**getSide**/**getRadius**), area (**getArea**) and perimeter (**getPerimeter**) and should override method **toString** from class **Object**.

The class **Square** should contain methods **getInscribedCircle** and **getCircumscribedCircle** returning objects of class **Circle** corresponding to circles inscribed in and circumscribed about *this* square. Similarily, the class **Circle** should contain methods **getInscribedSquare** and **getCircumscribedSquare** returning objects of class **Square** corresponding to squares inscribed in and circumscribed about *this* circle.

Add constructors to both classes: in class **Square** a constructor taking an object of class **Circle** and constructing a square with the same area as the given circle, and analogous constructor in class **Circle** taking an object of class **Square**.

Class **Circle** should define *static* function

```
public static Square[] getSquares(Circle[] arr)
```

which takes an array of references to objects of type **Circle** and creates and returns an array of objects of type **Square**, which have the same area as the corresponding circles.

In the **main** function *of another class* (e.g., **Main**) create a few objects of both classes and test the constructors and methods. For example, the following program

download *SquareCirc.java*

```
public class SquareCirc {
    public static void main (String[] args) {
        Square[] sqs = {new Square(2), new Square(1),
                    new Square(3), new Square(2)};
        for (Square s : sqs)
            System.out.print(s + " ");

        System.out.print("\nAreas: ");
        for (Square s : sqs)
            System.out.print(s.getArea() + " ");
        System.out.print("\nPerimeters: ");
        for (Square s : sqs)
            System.out.print(s.getPerimeter() + " ");

        Circle[] crs = {new Circle(2), new Circle(1),
                        new Circle(3), new Circle(2)};
        Square[] squares = Circle.getSquares(crs);
        System.out.println( "\nPerimeters of squares " +
                            "from circles: ");
        for (Square s : squares)
            System.out.printf("%.2f ", s.getArea());
        System.out.println();
    }
}
```

should print

```
Square[2.0] Square[1.0] Square[3.0] Square[2.0]
Areas: 4.0 1.0 9.0 4.0
Perimeters: 8.0 4.0 12.0 8.0
Perimeters of squares from circles:
12.57 3.14 28.27 12.57
```

NOTE: The value of π is available as Math.PI; square root of x can be calculated as Math.sqrt(x).

Problem 3

Create two classes – **Person** and **Car**. Class **Person** has two fields, name (**String**) and car (**Car**), and class **Car** has fields make (**String**) and price (**int**). In both classes all fields are private, so appropriate *getters* will probably be needed. Both classes override method **toString**.

Write two *static* member functions in class **Person**:

- **getCars** which takes an array of references to persons (persons have cars!), creates and returns an array of cars owned by the persons;
- **expensiveCars** which takes an array of persons and a minimum price (**int**), creates and returns an array of cars possessed by the persons, but only those cars with prices not less than the given minimum (the function returns **null** if there is no car which meets the criterion).

In function **main** (in a separate class) create an array of persons (owning cars!) and test both functions.
