Problem 1

Consider the following Java code:

class Transport {
    public static void main(String args[]) {
        Vehicle vehicle = new Vehicle();
        vehicle.print();
    }
}

class Vehicle {
    public Vehicle (double _milesPerGallon) {
        milesPerGallon = _milesPerGallon;
    }
    public void print() {
        System.out.println("MPG: "+milesPerGallon);
    }

    private double milesPerGallon = 10;
}

No, the code does not compile. Notice that the Vehicle class does not have a default constructor, but rather requires a double for the constructor. Thus, the call Vehicle vehicle = new Vehicle(); will generate an error during compilation. To correct the code, we would call the constructor with a double argument, e.g. Vehicle vehicle = new Vehicle(12.3); (our car is a big ugly SUV with bad gas mileage).
Problem 2

Consider the following modifications to our Java program:

class Transport {
    public static void main(String args[]) {
        Car car = new Car("Volkswagen");
        car.print();
    }
}

class Vehicle {
    public void print() {
        System.out.println("MPG: " + milesPerGallon);
    }
    private double milesPerGallon = 10;
}

class Car extends Vehicle {
    public Car(String _make) {
        make = _make;
    }
    public void print() {
        System.out.println("Make: " + make);
    }
    private String make;
}

Yes, the code does compile without error. The output is:
Make: Volkswagen.
Problem 3

Consider the following modifications to our Java program:

class Transport {
    public static void main(String args[]) {
        Car car = new Car("Volkswagen");
        car.setMilesPerGallon(26.9);
        car.print();
    }
}

class Vehicle {
    public void print() {
        System.out.println("MPG: "+milesPerGallon);
    }
    protected void setMilesPerGallon (double mpg) {
        milesPerGallon = mpg;
    }
    private double milesPerGallon = 10;
}

class Car extends Vehicle {
    public Car (String _make) {
        make = _make;
    }
    public void print() {
        System.out.println("Make:");
        System.out.println("MPG: "+milesPerGallon);
    }
    private String make;
}

No, the code does not compile. The Car class extends Vehicle. Since the setMilePerGallon method in Vehicle is protected, it is inherited by the Car class and the call to car.setMilesPerGallon is fine. However, the print method in the Car class tries to output the milesPerGallon double. But milesPerGallon is declared as a private double in the Vehicle class and therefore is not inherited by Car. Therefore, this line to output the milesPerGallon fails compilation. We can change the milesPerGallon variable declaration in Vehicle to be protected or public to fix the problem.
Problem 4

Consider the following modifications to our Java program:

class Transport {
    public static void main(String args[]) {
        Vehicle vehicle = new Vehicle();
        vehicle.setMilesPerGallon(26.9);
        Vehicle vehicle2 = (Vehicle)vehicle.clone();
        vehicle.setMilesPerGallon(22.1);
        vehicle.print();
        vehicle2.print();
    }
}

class Vehicle implements Cloneable {
    public Object clone() {
        try {
            Vehicle v = (Vehicle)super.clone();
            return v;
        } catch (CloneNotSupportedException e) {
            return null;
        }
    }

    public void print() {
        System.out.println("MPG: " + milesPerGallon);
    }
    protected void setMilesPerGallon (double mpg) {
        milesPerGallon = mpg;
    }
    protected double milesPerGallon = 10;
}

Yes, the code compiles. After the call to clone(), the vehicle and vehicle2 objects contain the same data. However, the call to vehicle.setMilesPerGallon(22.1); changes vehicle. When printing the two, the output of the program is:
MPG: 22.1
MPG: 26.9.