Lösningsförslag

Uppgift 1

```
import java.awt.Rectangle;
public class ImprovedRectangle extends Rectangle {
    public ImprovedRectangle(int width, int height) {
        super(width, height);
    }
    public ImprovedRectangle(int x, int y, int width, int height) {
        super(x, y, width, height);
    }
    public int getArea() {
        return (int) (getWidth() * getHeight());
    }
    public int getCircumference() {
        return (int) (2*(getWidth() + getHeight()));
    }
}//ImprovedRectangle
```

Uppgift 2

Ja, vi borde definierat ett gränssnitt

```
public interface Countable {
    public void count();
    public void reset();
    public int getValue();
}
```

och låtit klasserna Counter, BoundedCounter och ChainedCounter implementerat detta gränssnitt enligt:

```
public class Counter implements Countable { /*same code as before*/ }
public class BoundedCounter implements Countable { /*same code as before*/ }
public class ChainedCounter implements Countable { /*same code as before*/ }
```

Detta ger fördelen att datatyperna Counter, BoundedCounter och ChainedCounter alla är subtyper till datatypen Countable. Detta gör det möjligt att deklarera referensvariabler av datatypen Countable och tilldela dessa variabler referenser till objekt av datatypen Counter, BoundedCounter eller ChainedCounter. Används denna teknik blir alltså instanser av dessa tre klasser utbytbara.

Man kan också tänka sig att låta BoundedCounter utöka Counter och ChainedCounter utöka BoundedCounter:

```
public class Counter implements Countable {
    protected int value;
    /*same code as before*/
}//Counter

public class BoundedCounter extends Counter  {
    protected int modulus;
    public BoundedCounter(int modulus) {
        super();
        this.modulus = modulus;
    }//constructor
    public void count() {
        value = (value + 1) % modulus;
    }//count
}//BoundedCounter

public class ChainedCounter extends BoundedCounter {
    private Counter next;
    public ChainedCounter(int init, int modulus, Counter next) {
        super(modulus);
        this.value = init;
         this.next = next;
    }//constructor

    public void count() {
        value = (value + 1) % modulus;
        if (value == 0 && next != null)
              next.count();
    }//count
}// ChainedCounter
```

Uppgift 3

```
import java.awt.*;
public class GeometricObject {
    private Color color;
    private Point position;

    protected GeometricObject(){
        position = new Point(0,0);  //placeras i punkten  (0,0)
        color = Color.WHITE;       //ges färgen vit
    }
    protected GeometricObject(Point position, Color color){
        this.position = position;
        this.color = color;
    }
    public void setColor(Color color) {
        this.color = color;
    }
    public Color getColor() {
        return color;
    }

    public void setPosition(Point position) {
        this.position = position;
    }
    public Point getPosition() {
        return position;
    }
    public void move(Point position) {
        this.position = position;
    }
    public String toString() {
        return "Color: " + color + "\nPosition: " + position;
    }
}// GeometricObject
```

```
import java.awt.*;
public class Rectangle extends GeometricObject {
    private double width;
    private double length;
    public Rectangle() {
        this(1.0, 1.0);       //ges bredden 1 och längden 1
    }
    public Rectangle(double width, double length){
        this.width = width;
        this.length = length;
    }
    public Rectangle(double width, double length, Point position, Color color){
        super(position, color);
        this.width = width;
        this.length = length;
    }

    public void setWidth(double width) {
        this.width = width;
    }

    public double getWidth() {
        return width;
    }

    public void setLength(double length) {
        this.length = length;
    }

    public double getLength() {
        return length;
    }

    public double findArea() {
        return width*length;
    }
    public double findPerimeter() {
        return 2*(width + length);
    }

    public String toString() {
        return super.toString() + "\nWidth: " + width + "\nLength: " + length;
    }
}// Rectangle
```

```
import java.awt.*;
public class Circle extends GeometricObject {
    private double radius;
    public Circle(){
        this.radius = 1.0;
    }
    public Circle(double radius){
        this.radius = radius;
    }
    public Circle(double radius, Point position, Color color){
        super(position, color);
        this.radius = radius;
    }
    public void setRadius(double radius) {
        this.radius = radius;
    }

    public double getRadius() {
        return radius;
    }
    public double findArea() {
        return Math.PI*radius*radius;
    }
    public double findPerimeter() {
        return 2*radius*Math.PI;
    }
    public String toString() {
        return super.toString() + "\nRadius: " + radius;
    }
}// Circle
```

```
import java.awt.*;
public class Cylinder extends Circle {
    private double height;
    public Cylinder(){
        this.height = 1.0;
    }
    public Cylinder(double radius, double height){
        super(radius);
        this.height = height;
    }
    public Cylinder(double radius, double height, Point position, Color color){
        super(radius, position, color);
        this.height = height;
    }
    public void setHeight(double height) {
        this. height = height;
    }

    public double getHeight () {
        return height;
    }
    public double findArea() {
        return 2* super.findArea() +  height*findPerimeter();
    }
    public double findVolym() {
        return super.findArea() * height;
    }
    public String toString() {
        return super.toString() + "\nHeight: " + height;
    }
}// Cylinder
```

Uppgift 4

a)

```
//before: people != null
public static ArrayList<String> bornThisYear(ArrayList<Person> people, int year) {
    ArrayList<String> res = new ArrayList<String>();
    for (Person p : people) {
        if (p.getYear() == year)
            res.add(p.getName());
    }
    return res;
}//bornThisYear
```

b)

```
//before: people != null
public static void removeNames(ArrayList<Person> people,  ArrayList<String>  names) {
    int pos = 0;
    while (pos < people.size()) {
        if (isInList(names, people.get(pos).getName()))
            people.remove(pos);
        else
            pos++;
    }
}//removeNames

//before: names != null
private static boolean isInList(ArrayList<String>  names, String name) {
    for (String elem: names) {
        if (elem.equals(name))
            return true;
    }
    return false;
}//isInList
```

Diskutera/förklara varför inte följande ”lösningsförslag” fungerar?

```
//before: people != null
public static void removeNames(ArrayList<Person> people,  ArrayList<String>  names) {
    for (int pos = 0; pos < people.size(); pos++) {
        if (isInList(names, people.get(pos).getName())) {
            people.remove(pos);
        }
    }
}//removeNames
```

Uppgift 5

```
public enum SuitColor {
    RED, BLACK;
}//SuitColor

public enum Suit {
    HEARTS(SuitColor.RED), SPADES(SuitColor.BLACK), DIAMONDS(SuitColor.RED),
    CLUBS(SuitColor.BLACK);
    private SuitColor color;
    public Suit(SuitColor color) {
        this.color = color;
    }
    public SuitColor getColor() {
        return color;
    }
}//Suit
```

Uppgift 6

```
public interface Superpower {
    public void fly();
    public void heal();
    public void rescue();
}//Superpover

public class Human {
    public void eat(int calories) {
    }
    public void drink(int volyme) {
    }
    public void sleep(int minutes) {
    }
}//Human

public Superhero extends Human interface Superpower {
    public void fly() {
    }
    public void heal() {
    }
    public void rescue() {
    }
}//Superhero
```