

# Lösningförslag, omkontrollskrivning PTDC

2013-08-22

```
1. public class Tree {
    private String species;
    private double radius;
    private double x;
    private double y;

    public Tree(String species) {
        this.species = species;
        radius = TreeSizes.getRadius(species);
        x = Double.MIN_VALUE;
        y = Double.MIN_VALUE;
    }

    public void placeAt(double newX, double newY) {
        x = newX;
        y = newY;
    }

    public String getSpecies() {
        return species;
    }

    public double getX() {
        return x;
    }

    public double getY() {
        return y;
    }

    public double getRadius() {
        return radius;
    }

    public boolean overlaps(Tree t) {
        double dist = Math.hypot(x - t.x, y - t.y);
        return dist < radius + t.radius;
    }
}
```

---

```
2. public class Park {
    private String name;
    private double width;
    private double height;
    private ArrayList<Tree> trees;
    private static Random rand = new Random();

    public Park(String name, double width, double height) {
        this.name = name;
        this.width = width;
        this.height = height;
        trees = new ArrayList<Tree>();
    }

    public String getName() {
        return name;
    }

    public double getWidth() {
        return width;
    }

    public double getHeight() {
        return height;
    }

    public void placeTree(Tree t) {
        boolean placeOk = false;
        do {
            double x = randReal(t.getRadius(), width - t.getRadius());
            double y = randReal(t.getRadius(), height - t.getRadius());
            t.placeAt(x, y);
            int i = 0;
            while (i < trees.size() && !trees.get(i).overlaps(t)) {
                i++;
            }
            if (i == trees.size()) {
                placeOk = true;
            }
        } while (!placeOk);
        trees.add(t);
    }

    public Tree[] getTrees() {
        Tree[] tArray = new Tree[trees.size()];
        for (int i = 0; i < tArray.length; i++) {
            tArray[i] = trees.get(i);
        }
        return tArray;
    }

    private static double randReal(double a, double b) {
        return a + rand.nextDouble() * (b - a);
    }
}
```

---

---

```
3. public class ParkPlanner {
    private Park park;

    public ParkPlanner(Park park) {
        this.park = park;
    }

    public void placeTrees(ArrayList<Tree> treeList) {
        while (!treeList.isEmpty()) {
            Tree largest = treeList.get(0);
            for (int i = 1; i < treeList.size(); i++) {
                if (treeList.get(i).getRadius() > largest.getRadius()) {
                    largest = treeList.get(i);
                }
            }
            treeList.remove(largest);
            park.placeTree(largest);
        }
    }

    public void print() {
        Tree[] trees = park.getTrees();
        for (int i = 0; i < trees.length; i++) {
            Tree t = trees[i];
            System.out.println(t.getSpecies() + " "
                               + t.getX() + " " + t.getY());
        }
    }

    public void draw(double scale) {
        int width = (int) (scale * park.getWidth());
        int height = (int) (scale * park.getHeight());
        Window w = new Window(park.getName(), width, height);
        Tree[] trees = park.getTrees();
        for (int i = 0; i < trees.length; i++) {
            Tree t = trees[i];
            int x = (int) (t.getX() * scale);
            int y = (int) (t.getY() * scale);
            int radius = (int) (t.getRadius() * scale);
            w.drawCircle(x, y, radius);
        }
    }
}

4. public class ParkMain {
    public static void main(String[] args) {
        System.out.println("Skriv parkens namn och storlek");
        Scanner scan = new Scanner(System.in);
        String name = scan.next();
        int width = scan.nextInt();
        int height = scan.nextInt();
        Park park = new Park(name, width, height);
        scan.close();
    }
}
```

---

```
ArrayList<Tree> treeList = new ArrayList<Tree>();
try {
    scan = new Scanner(new File("trees.txt"));
} catch (FileNotFoundException e) {
    System.err.println("Cannot open trees.txt");
    System.exit(1);
}
while (scan.hasNext()) {
    String species = scan.next();
    treeList.add(new Tree(species));
}
scan.close();

ParkPlanner planner = new ParkPlanner(park);
planner.placeTrees(treeList);
planner.print();
planner.draw(5);
}
}
```