Санкт-Петербургский национальный исследовательский университет

информационных технологий, механики и оптики

Лабораторная работа №2

150/GET

**4100**

Дымов

Шамуков

Санкт-Петербург

2012

Оглавление

Задание 3

Исходный код 4

Index.jsp 4

style.css 8

ShapeUtil 8

ChartModel 9

Position 10

Strategy 11

Util 11

BaseShape 12

CircleSectorShape 12

IShape 13

MultipleShape 13

RectangleShape 14

TriangleShape 14

Вывод 15

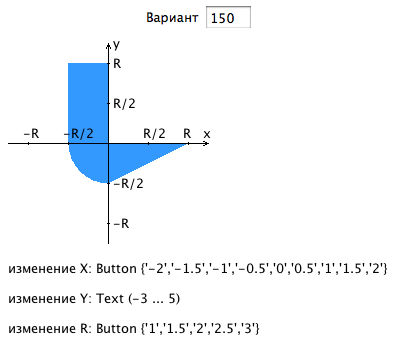
# Задание

Создать JSP-страницу, которая формирует HTTP-запрос на проверку попадания точки в заданную область на координатной плоскости и осуществляет обработку этого запроса.

Страница должна содержать:

1. "Шапку", содержащую ФИО студента, номер группы и номер варианта.
2. Форму, отправляющую данные на сервер. Форма должна использовать метод HTTP-запроса в соответствии с вариантом из работы №4.
3. Набор полей для задания координат точки и радиуса области в соответствии с вариантом задания.
4. Картинку, изображающую область на координатной плоскости в соответствии с номером варианта. Если радиус области установлен, клик курсором мыши по этой картинке должен обрабатываться Javascript-функцией, определяющей координаты точки, по которой кликнул пользователь. В противном случае, после клика по картинке должно выводиться сообщение о невозможности определения координат точки.
5. Код на языке Java, осуществляющий проверку попадания точки в область.
6. Таблицу со списком результатов предыдущих проверок.

Разработанную JSP-страницу необходимо развернуть на сервере приложений GlassFish.



# Исходный код

## Index.jsp

<%@ taglib uri="http://java.sun.com/jsp/jstl/core"

prefix="c" %>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<%@ page import="java.util.ArrayList" %>

<%@ page import="model.ChartModel" %>

<%@ page import="static model.ShapeUtil.getLab2Shape" %>

<html >

<script type="text/javascript">

var isXValid=false;

var isYValid=false;

var isRValid=false;

var R;

function onRButtonClick(e) {

document.getElementById("r").value = document.getElementById(e).value;

if (document.getElementById(e).value > 0) {

R = document.getElementById(e).value;

isRValid = true;

if (isXValid && isYValid && isRValid) {

enableSubmit();

}

} else {

isRValid=false;

disableSubmit()

}

}

function onXButtonClick(e) {

var newValue = document.getElementById(e).value;

document.getElementById("x").value = newValue;

if (newValue.match(/^-?\d+([.|,]\d+)?$/)) {

isXValid=true;

if (isXValid && isYValid && isRValid) {

enableSubmit();

}

} else {

isXValid=false;

disableSubmit();

}

}

function onYChange(e) {

if (e.match(/^-?\d+([.|,]\d+)?$/) && e > -3 && e < 5) {

isYValid=true;

if (isXValid && isYValid && isRValid) {

enableSubmit();

}

} else {

isYValid=false;

disableSubmit();

}

}

function enableSubmit() {

document.getElementById("submit").disabled = false;

}

function disableSubmit() {

document.getElementById("submit").disabled = true;

}

function getPoint(event) {

if (!isRValid)

alert("Please set Radius first!");

else {

var pos\_x = event.offsetX?(event.offsetX):event.pageX-document.getElementById("pointer-div").offsetLeft;

var pos\_y = event.offsetY?(event.offsetY):event.pageY-document.getElementById("pointer-div").offsetTop;

pos\_x = pos\_x \* R \* 2 / 170 - R;

pos\_y = R - pos\_y \* R \* 2 / 168;

document.pointform.form\_x.value = pos\_x;

document.pointform.form\_y.value = pos\_y;

isXValid=true;

isYValid=true;

if (isXValid && isYValid && isRValid)

enableSubmit();

document.enterform.x.value = pos\_x;

document.enterform.y.value = pos\_y;

}

}

</script>

<head>

<title>Lab 6</title>

<link href="style.css" rel="stylesheet" type="text/css">

Lab 6, Shamukov A. & Dymov E. at 4100

</head>

<body>

<p>Is point inside figure?</p>

<form method="GET" action="/lab2\_war\_exploded/" enctype="multipart/form-data" name="enterform">

<table id="input-table">

<tr>

<td class="first-td">

x:

</td>

<td>

<input type=text name="x" id="x" size=5 onkeypress="return event.keyCode!=13" readonly="true" >

</td>

<td>

<input type="button" value="-2" id="x1" onclick="onXButtonClick(this.id)">

<input type="button" value="-1.5" id="x2" onclick="onXButtonClick(this.id)">

<input type="button" value="-1" id="x3" onclick="onXButtonClick(this.id)">

<input type="button" value="-0.5" id="x4" onclick="onXButtonClick(this.id)">

<input type="button" value="0" id="x5" onclick="onXButtonClick(this.id)">

<input type="button" value="0.5" id="x6" onclick="onXButtonClick(this.id)">

<input type="button" value="1" id="x7" onclick="onXButtonClick(this.id)">

<input type="button" value="1.5" id="x8" onclick="onXButtonClick(this.id)">

<input type="button" value="2" id="x9" onclick="onXButtonClick(this.id)">

</td>

</tr>

<tr>

<td class="first-td">

y:

</td>

<td>

<input type=text name="y" size=5 onkeypress="return event.keyCode!=13" onchange="onYChange(this.value)">

</td>

<td>

</td>

</tr>

<tr>

<td class="first-td">

r:

</td>

<td>

<input type=text name="r" id="r" size=5 onkeypress="return event.keyCode!=13" readonly="true" >

</td>

<td>

<input type="button" value="1" id="r1" onclick="onRButtonClick(this.id)">

<input type="button" value="1.5" id="r2" onclick="onRButtonClick(this.id)">

<input type="button" value="2" id="r3" onclick="onRButtonClick(this.id)">

<input type="button" value="2.5" id="r4" onclick="onRButtonClick(this.id)">

<input type="button" value="3" id="r5" onclick="onRButtonClick(this.id)">

</td>

</tr>

<tr>

<td colspan="3">

<br/>

<input type=submit value="Send data to server" id="submit" disabled="true">

</td>

</tr>

</table>

</form>

<form name="pointform" method="post">

<div id="pointer-div" onclick="getPoint(event)"></div>

<p>

You clicked at:

<br>

<input type="text" name="form\_x" size="4" readonly="true" />

<input type="text" name="form\_y" size="4" readonly="true" />

</p>

</form>

<table id="res-table" border="2px">

<%= processRequestAndGetLog(request, out) %>

</table>

</body>

</html>

<%!

private final StringBuilder resultLog = new StringBuilder();

public String formatData(float x, float y, float r, boolean isContains) {

return String.format("<tr><td>%s</td><td>%s</td><td>%s</td><td>%s</td></tr>",

String.valueOf(x),

String.valueOf(y),

String.valueOf(r),

String.valueOf(isContains));

}

private String processRequestAndGetLog(final HttpServletRequest request, final JspWriter out) {

try {

if (request != null) {

final ChartModel chartModel = new ChartModel(getLab2Shape());

final float x = Float.valueOf(request.getParameter("x"));

final float y = Float.valueOf(request.getParameter("y"));

final float r = Float.valueOf(request.getParameter("r"));

chartModel.setCurrentX(x);

chartModel.setCurrentY(y);

chartModel.setRadius(r);

final boolean containsLiteral = chartModel.containsCurrentPosition();

resultLog.append(formatData(x, y, r, containsLiteral));

}

} catch (final NullPointerException e) {

//Do nothing

}

return resultLog.toString();

}

%>

## style.css

body {

font-family: verdana, tahoma, sans-serif;

color: #333366;

text-align: center;

margin-top: 50px;

}

#pointer-div {

background-image:url('var150\_cropped.jpg');

width:168px;

height:170px;

margin-left:auto;

margin-right:auto;

}

#res-table {

margin-left:auto; margin-right:auto; width:25%

}

#input-table {

margin-left:auto; margin-right:auto; width:45%

}

first-td {

width:10%

}

## ShapeUtil

package model;

import model.shape.\*;

import model.stuff.Position;

import model.stuff.Strategy;

public class ShapeUtil {

public static BaseShape getLab2Shape() {

final CircleSectorShape circleSector = new CircleSectorShape(new Strategy() {

public Position compute(float radius) {

return new Position(- radius / 2, -radius / 2);

}

});

final RectangleShape rectangle = new RectangleShape(new Strategy() {

public Position compute(float radius) {

return new Position(- radius / 2, radius);

}

});

final TriangleShape triangle = new TriangleShape(new Strategy() {

public Position compute(float radius) {

return new Position(radius, - radius / 2);

}

});

final MultipleShape shape = new MultipleShape();

shape.addShape(circleSector);

shape.addShape(rectangle);

shape.addShape(triangle);

shape.setRadius(1);

return shape;

}

}

## ChartModel

package model;

import model.shape.BaseShape;

import model.stuff.Position;

import view.IView;

import java.util.ArrayList;

import java.util.List;

public class ChartModel {

private List<IView> myViews = new ArrayList<IView>();

Position myCurrentPosition = new Position(0, 0);

BaseShape myShape;

public ChartModel(BaseShape shape) {

myShape = shape;

}

protected void updateViews() {

for (IView view : myViews) {

view.updateView();

}

}

public void addView(IView view) {

myViews.add(view);

}

public void setCurrentX(float x) {

myCurrentPosition.setX(x);

updateViews();

}

public void setCurrentY(float y) {

myCurrentPosition.setY(y);

updateViews();

}

public void setRadius(float r) {

myShape.setRadius(r);

updateViews();

}

public Position getCurrentPosition() {

return myCurrentPosition;

}

public boolean containsCurrentPosition() {

return myShape.contains(getCurrentPosition());

}

}

## Position

package model.stuff;

public class Position {

private float myX;

private float myY;

public Position(float x, float y) {

myX = x;

myY = y;

}

public float getX() {

return myX;

}

public float getY() {

return myY;

}

public void setX(float x) {

myX = x;

}

public void setY(float y) {

myY = y;

}

@Override

public String toString() {

return "x = " + myX + "\t y = " + myY;

}

}

## Strategy

package model.stuff;

public interface Strategy {

Position compute(float radius);

}

## Util

package model.stuff;

public class Util {

private static boolean isBetweenCoordinates(float searchedValue, float value1, float value2) {

float minValue = Math.min(value1, value2);

float maxValue = Math.max(value1, value2);

return searchedValue >= minValue && searchedValue <= maxValue;

}

public static boolean isBetweenPositions(Position searchedPosition, Position p1, Position p2) {

return isBetweenCoordinates(searchedPosition.getX(), p1.getX(), p2.getX())

&& isBetweenCoordinates(searchedPosition.getY(), p1.getY(), p2.getY());

}

public static float getDistance(Position p1, Position p2) {

float distanceX = p1.getX() - p2.getX();

float distanceY = p1.getY() - p2.getY();

return (float) Math.sqrt(distanceX \* distanceX + distanceY \* distanceY);

}

}

## BaseShape

package model.shape;

import model.stuff.Position;

import model.stuff.Strategy;

public abstract class BaseShape implements IShape {

private final Position myFixedPosition = new Position(0, 0);

private Strategy myStrategy;

private float myRadius;

public BaseShape(Strategy strategy) {

myStrategy = strategy;

}

public float getRadius() {

return myRadius;

}

public void setRadius(float radius) {

myRadius = radius;

}

public Position getUnfixedPosition() {

return myStrategy.compute(getRadius());

}

public Position getFixedPosition() {

return myFixedPosition;

}

}

## CircleSectorShape

package model.shape;

import model.stuff.Position;

import model.stuff.Strategy;

import model.stuff.Util;

public class CircleSectorShape extends BaseShape {

public CircleSectorShape(Strategy circleRadiusStrategy) {

super(circleRadiusStrategy);

}

private float getCircleRadius() {

return Math.abs(getUnfixedPosition().getX() - getFixedPosition().getX());

}

public boolean contains(Position position) {

Position maxPosition = getUnfixedPosition();

if (!Util.isBetweenPositions(position, getFixedPosition(), maxPosition)) {

return false;

}

float positionRadius = Util.getDistance(position, getFixedPosition());

return positionRadius <= getCircleRadius();

}

}

## IShape

package model.shape;

import model.stuff.Position;

public interface IShape {

void setRadius(float r);

boolean contains(Position position);

}

## MultipleShape

package model.shape;

import model.stuff.Position;

import java.util.HashSet;

import java.util.Set;

public class MultipleShape extends BaseShape {

private Set<IShape> myParts = new HashSet<IShape>();

public MultipleShape() {

// position strategy can't be used

super(null);

}

public void addShape(IShape shape) {

myParts.add(shape);

}

@Override

public void setRadius(float radius) {

super.setRadius(radius);

for (IShape part : myParts) {

part.setRadius(radius);

}

}

public boolean contains(Position position) {

for (IShape part : myParts) {

if (part.contains(position)) {

return true;

}

}

return false;

}

public Set<IShape> getShapes() {

return myParts;

}

}

## RectangleShape

package model.shape;

import model.stuff.Position;

import model.stuff.Strategy;

import model.stuff.Util;

public class RectangleShape extends BaseShape {

public RectangleShape(Strategy unfixedPositionStrategy) {

super(unfixedPositionStrategy);

}

public boolean contains(Position position) {

return Util.isBetweenPositions(position, getFixedPosition(), getUnfixedPosition());

}

}

## TriangleShape

package model.shape;

import model.stuff.Position;

import model.stuff.Strategy;

import model.stuff.Util;

public class TriangleShape extends BaseShape {

public TriangleShape(Strategy strategy) {

super(strategy);

}

private float getCoefficientSlope() {

return -getUnfixedPosition().getY() / getUnfixedPosition().getX();

}

private float getFreeTerm() {

return getUnfixedPosition().getY();

}

private Position getPositionByX(float x) {

float y = getCoefficientSlope() \* x + getFreeTerm();

return new Position(x, y);

}

public boolean contains(Position position) {

if (!Util.isBetweenPositions(position, getFixedPosition(), getUnfixedPosition())) {

return false;

}

Position positionToCompare = getPositionByX(position.getX());

float searchedDistance = Util.getDistance(getFixedPosition(), position);

float maxDistance = Util.getDistance(getFixedPosition(), positionToCompare);

return searchedDistance <= maxDistance;

}

}

# Вывод

Проделав лабораторную работу, мы познакомились с основами JSP.