|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 입사지원서 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **입사구분** | | | | 신입 | | | | | | | | | | | | | **응시부문** | | | | | | | | | 소프트웨어개발 | | | | | | | | | | | **희망급여** | | | | | | 0000만원 | | | |
| **긴급연락처** | | | | 1) 휴대폰: 000-0000-0000 2) E-Mail: | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **인**  **적**  **사**  **항** | **성 명** | | | | | **한글** | | | | |  | | | | | | | | | | | | | | | **漢字** | | | |  | | | | | | | | | | | |  | | | | |
| **Name** | | | | |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 주민등록번호 | | | | |  | |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | |  | |  | |  |  | |  |  | | | |  | |  | |  | |  |  | | |  | |  | (만 00세) | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **E-Mail** | | | | |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **연락처** | | | | |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **현주소** | | | | |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **학**  **력**  **사**  **항** | **졸 업 년 월** | | | | | | **학 교 명** | | | | | | | | | | | | | | | | | | | | | **전 공** | | | | | | | | | | | **소재지** | | | | | | **성 적** | |
| 0000년 00월 | | | | | |  | | | | | | | | | | | | | | | | | | | | | 00/00 | | | | | | | | | | |  | | | | | | 0.00/4.5 | |
| 0000년 00월 | | | | | |  | | | | | | | | | | | | | | | | | | | | |  | | | | | | | | | | |  | | | | | |  | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **프**  **로**  **젝**  **트** | | 수행 프로젝트 및 구현내용 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **구분** | **프로젝트명** | | | | | | | | | **년도/**  **개발기간** | | | | | | | | | | **사용언어/**  **개발환경** | | | | | | | | | | **구 현** | | | | | | | | | | | | | | |
| 팀별 | Java Game Programing | | | | | | | | | 0000.00 | | | | | | | | | | Java/  Windows xp | | | | | | | | | | **Java JApplet, Server/Client, UI구현** | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **기**  **술**  **능**  **력** | | **자료구조** | | | **알고리즘** | | | | | | | | | **C/C++** | | | | | | | | | | **JAVA** | | | | | | | **안드로이드** | | | | | | | **포토샵** | | | | | | **데이터**  **베이스** | | |
| 상 | | | 상 | | | | | | | | | 중 | | | | | | | | | | 상 | | | | | | | 중 | | | | | | | 중 | | | | | | 중 | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **경**  **력**  **사**  **항** | | **근무 기간** | | | | | | | | **근 무 처** | | | | | | | | | | | **직 위** | | | | | | | | | | | | | **담당 업무** | | | | | | | **퇴직 사유** | | | | | |
| 00년0월~00년0월 | | | | | | | |  | | | | | | | | | | |  | | | | | | | | | | | | |  | | | | | | |  | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **자**  **격**  **사**  **항** | | **종류** | | | | | | | **발행처** | | | | | | | | | | | **취득일** | | | | | | | | | | | | | | **외**  **국**  **어** | | **언어** | | | | **능력** | | | | | | **공인시험** |
|  | | | | | | |  | | | | | | | | | | |  | | | | | | | | | | | | | |  | | | |  | | | | | |  |
|  | | | | | | |  | | | | | | | | | | |  | | | | | | | | | | | | | | **O A**  **능력** | | **엑셀** | | | | **워드** | | | | | | **프레젠테이션** |
|  | | | | | | |  | | | | | | | | | | |  | | | | | | | | | | | | | |  | | | |  | | | | | |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **개**  **인**  **사**  **항** | **신장** | **체중** | **혈액형** | **특기** | **취미** | **시력** | **병**  **역** | **구 분** |
|  |  |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **가**  **족**  **사**  **항** | **관계** | **성명** | **나이** | **직업** | **동거여부** |
|  |  |  |  |  |
|  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 자기소개서 | | | | | |
| **입사구분** | 신입 | **응시부문** | 소프트웨어개발 | **희망급여** | 0000만원 |
| **긴급연락처** | 1) 휴대폰: 000-0000-0000 2) E-Mail: | | | | |
|  | | | | | |
| **축구계의 멀티플레이어 박지성!**  **프로그래밍 세계의 멀티플레이어가 되고 싶은 홍길동!!**  저는 어떤 곳에서든 묵묵히 열심히 뛰는 박지성이라는 축구선수를 좋아합니다.  맨체스터 유나이티드의 퍼거슨 감독은 자신의 전술을 가장 잘 이해하고 성실한 선수로 박지성을 꼽는다고 합니다. 그 이유는 어느 위치에서든 열심히 노력하는 자세와 변화에 두려워하지 않는 마음가짐 때문이라고 생각합니다.  **“’노력’ 이라는 진심은 통한다!”**  한때 저는 ‘000000㈜’라는 인력 아웃소싱 회사의 인사관리 담당자였습니다. 조금 더 일찍 움직이고 조금 더 늦게 퇴근했습니다. 계약직이지만 업무에 열정을 가지고 열심히 일하던 중 ‘000000’ 중간관리자 업무평가에서 전국 1위를 하게 되었고 그 결과 3개월이라는 짧은 기간 내에 정규직으로 전환될 수 있었습니다. 회사 내에서도 좋은 이미지를 쌓아 저와 같은 사례에서 사원을 채용하는 현상까지 나타나게 되었습니다.  제가 그 업무에 특별한 재능이 있거나 능력이 있어서 좋은 평가를 받았다고는 생각하지 않습니다. 단지 제가 맡은 업무에 열정을 가지고 조금씩 노력해 나갔기 때문에 좋은 결과가 뒤따라 준 것이라 생각합니다.  **“책임감 + 열정 = 프로그래머”**  사랑하는 사람과 결혼을 하게 되었고 저를 꼭 닮은 아이까지 갖게 되었습니다. 그리고 곧, 책임감이 뒤따랐습니다. 그 책임감은 나를 알고, 나를 일으킬 수 있는 무언가를 찾기 시작했습니다.  어린 시절 저에게는 항상 컴퓨터가 있었습니다. 게임보다는 무엇인가를 만드는 것을 좋아했던 저는 gw-basic으로 수학문제도 풀고, 그림도 그릴 수 있었습니다. 지금 생각해 보면 정말 단순한 내용의 것들이었지만 그때는 이유 없이 즐거웠습니다. 저의 열정이 다른 곳에 있다는 것을 깨닫고 더 이상 일을 할 수가 없었습니다. 무작정 컴퓨터로 무엇인가를 만들고 싶었습니다. 우선 기초부터 시작해야만 했습니다. 그러던 중 ‘유진능력개발원’이라는 곳을 알게 되었고, 제 마음은 ‘엑센츄어’라는 곳을 향하기 시작했습니다.  **“코끼리 그림의 빡쎈츄어??”**  저는 ‘엑센츄어’라는 곳을 전혀 몰랐습니다. 그런데 그 곳에는 밥 먹는 시간 빼고 프로그래밍만 한다고 합니다. 이게 만약 사실이라면 저에게는 기회입니다. 밥 먹고 프로그래밍만 하고 싶기 때문입니다. 늦게 시작한 이 일을 남들과 똑같이 해서 어느 세월에 든든한 가장이 될 수 있겠습니까?  **“변화를 두려워 하지 말자!”**  미래에 대한 청사진이나 돈에 대한 욕심을 부렸다면 바로 실행에 옮기지 못했을 것입니다. 좋아하는 일을 하게 될지도 모른다는 설레임과 저를 믿어주는 가족이 있었기 때문에 가능한 일이었습니다.  귀사에 이력서를 쓰기까지는 불과 5개월이라는 시간 밖에 걸리지 않았습니다. 지금은 미완성이고 프로그래머라 말할 수 없지만, 저는 변해가고 있습니다. 앞으로 50년을 변해갈 것입니다.  누구나 프로그래머가 될 수 있을지 모릅니다. 하지만 그 일을 좋아하고 끝까지 성실히 해 낼 사람이 필요하시다면 저를 꼭 추천해 드리고 싶습니다. 너무 늦게 제가 좋아하는 일을 찾게 되었지만 가장 정확히 배우고 열정이 있는 성실함으로 귀사에 도움이 되는 인재가 되겠습니다. 기회를 주신다면 진심으로 최선을 다하겠습니다. | | | | | |

입사지원서상의모든기재사항은사실임을확인합니다.

0000년00월00일

작성자홍길동

|  |  |
| --- | --- |
| **수행프로젝트** | |
| **작품명** | **Java Tetris Game** |
| **개발기간** | **2012.05.10~06.20** |
| **개발환경** | **Java JDK 1.7.0, Windows XP, Eclipse IDE, illustrator cs4** |

**아래의 내용은 JAVA Game Programing 프로젝트로 만든 Tetris Game 소스 입니다. 키보드의 키를 입력 받아 블록이 움직이며 버튼을 클릭했을 시에 게임이 일시 정지 되는 효과, 그리고 게임 상황에 따라 삽입되는 배경음악과, 블록이 떨어질 자리를 미리 보여주는 고스트 블록 등을 구현하였습니다.**

**결과 화면:**

****

**소스 설명:**

**메인 메서드에서 Tetris클래스를 호출 합니다.**

**public** Tetris() {//Constructor

**super**("3Bear's Tetris"); //프레임제목설정

**this**.init();

**super**.setSize(mwidth+6, mheight+28);

Dimension screen = Toolkit.*getDefaultToolkit*().getScreenSize();

Dimension frm = **super**.getSize();

**int** xpos = (**int**)((screen.getWidth()/2) - (frm.getWidth()/2));

**int** ypos = (**int**)((screen.getHeight()/2) - (frm.getHeight()/2));

**super**.setLocation(xpos, ypos); //게임프레임의위치설정

**super**.setVisible(**true**); //프레임보이기

**super**.setResizable(**false**); //크기조절하지못하게함

**super**.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);

}

**위의 내용을 실행하면 비어있는 프레임이 생성되며 프레임에 배경 이미지와 버튼 이미지를 넣고 버튼을 클릭했을 경우에 게임이 실행되는 속성 등을 지정합니다.**

**publicvoid** init(){

(중략)

bg = **new** JLabel(**new** ImageIcon("images/back1.png")); bg.setBounds(0,0, mwidth, mheight);

**this**.Buttons(380,467,165,165,"images/start1.png","images/start2.png"); //버튼생성

**this**.add(bg, -1); //프레임에배경설정

**this**.add(gpn,0); //Game Panel 추가

**this**.add(prepn, 0); //Prepared Panel 추가

**this**.add(spn, 0); //Score Panel 추가

**this**.add(lpn, 0); //Level Panel 추가

**this**.add(tpn, 0); //Timer Panel 추가

**this**.add(button, 0); //Button 추가

}

**publicvoid** Buttons(**int** x, **int** y, **int** w, **int** h, String bpath, String press){

button = **new** JButton(**new** ImageIcon(bpath));

button.setBorderPainted(**false**);

button.setContentAreaFilled(**false**);

button.addActionListener(**this**); //버튼에액션추가

(중략)

}

**publicvoid** actionPerformed(ActionEvent e){

**if** (buttonCheck&&first){//게임을처음시작했고, 버튼체크가true일때실행(아직게임이진행되지는않음)

gpn.init(); //게임실행

**this**.Focus(); //게임보드에포커스를준다.

**this**.Buttons(380,467,165,165,"images/stop1.png","images/stop2.png");

}

**elseif** (buttonCheck&& !first){(중략) } //게임진행중, 일시정지중인상황에서다시게임을시작하고싶을경우

**위의 내용처럼 버튼에 기능을 부여하고 클릭했을 시에 게임이 진행되도록 설정합니다. 다음으로 게임 패널을 30\*30 픽셀에 가로10칸, 세로20칸의 크기로 나눕니다. 그리고 게임 스타트 시에는 thread가 동작하여 run()메서드가 작동합니다.**

**publicvoid** run() {

playSound("sound/startnormal.wav");

**while** (isEnd==0) {//꼭대기 자리에 도형이 없을 경우0

pre.setVariable(nextCol, nextType); //다음도형을미리보여주는 prePN클래스에다음도형의타입과색상을전달

pre.repaint();

**try** {

Thread.*sleep*(delay);

**while**(pause) {//게임 정지 시에 while문 안으로

clip.stop();

Thread.*sleep*(1000);

**if** (!pause) {

clip.start();

clip.loop(Clip.*LOOP\_CONTINUOUSLY*);

**break**;

}

}

}**catch** (InterruptedException e) { }

lBlocked=**false**;

//도형을새로움직여야하기때문에모두 false값으로초기화

rBlocked=**false**;

dBlocked=**false**;

tBlocked=**false**;

**if**(!isDrop()){

nowY[0]=nowY[0]+30;

nowY[1]=nowY[1]+30;

nowY[2]=nowY[2]+30;

nowY[3]=nowY[3]+30;

repaint(); //repaint를통해새로그려줌

}

spn.setScore(score); //현재점수를전달

lpn.setLevel(level); //현재레벨을전달

}

clip.stop();

playSound("sound/die.wav");

}

**위의 run()메서드 안에서 게임 동작이 이루어집니다. Thread 가 반복되며 꼭대기까지 블록이 쌓이지 않았는지 감시하며 블록이 쌓이지 않은 경우, 다음에 생성될 도형을 랜덤으로 지정해 우측 화면에 미리 보여줍니다. 그리고 블록의 Y좌표에 한 픽셀의 크기인 30을 더해서 한 칸씩 아래로 내려오는 효과를 부여합니다.**

**이후 블록이 아래에 떨어질 자리를 미리 보여주는 고스트 블록을 추가하였습니다. 고스트 블록은 미리 임시좌표를 생성하여 임시 Y좌표를 보드판의 제일 꼭대기에 위치시킨 후, 아래 while문을 통해 제일 윗줄부터 쌓인 블록이 있는지 없는지 검사를 하는데, 검사를 한 후에 바로 아래에 블록이 있거나 제일 밑바닥에 닿은 경우 그 자리에 고스트 블록을 그리는 방식으로 구현하였습니다.**

**publicvoid** ghostXY\_Turn() { //현재도형의타입에따라서임시좌표인 gtempX, gtempY값지정

gtempX[0]=nowX[0];gtempX[1]=nowX[1];

gtempX[2]=nowX[2];gtempX[3]=nowX[3];

**switch**(nowType) {// 현재블럭의타입과맞는case를찾아감!

**case**"longtype" : {

**if**(angle==1 || angle==3) {//90도, 270도회전

gtempY[0]=30;gtempY[1]=0;gtempY[2]=60;gtempY[3]=90;

**break**;

}**else**{//기본각도,180도회전

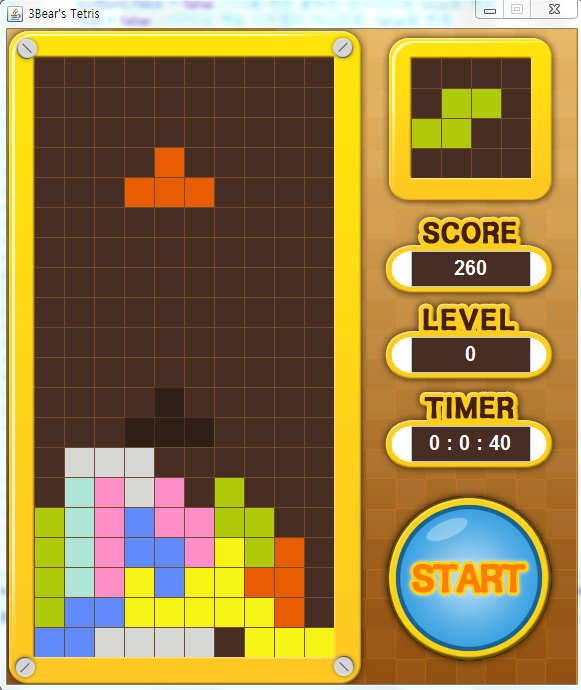
gtempY[0]=0;gtempY[1]=0;gtempY[2]=0;gtempY[3]=0;

**break**;

}

} (중략)

**위와 같은 방식으로 기본 도형의 위치와 모양에 따라 고스트 블록을 생성하여 아래에 그림자처럼 보여줍니다. 여기까지의 코드를 실행하면 아래 그림과 같습니다.**

****

**배경음악을 재생시키는 코드는 다음과 같습니다.**

**publicvoid** playSound(String filename){

**try** {

FileInputStream fis = **new** FileInputStream(fileName);

BufferedInputStream bs=**new** BufferedInputStream(fis);

AudioInputStream as =AudioSystem.*getAudioInputStream*(bs);

//FileInputStream에매개변수로전달되는음악파일을넣고 BufferedInputStream에파일의크기만큼의데이터를내부버퍼에저장시킴. AudioInputStream을이용해서데이터를입력받음.

**if**(fileName=="sound/startnormal.wav") {

clip = AudioSystem.*getClip*();

clip.open(as);

clip.start();

clip.loop(Clip.*LOOP\_CONTINUOUSLY*);

} **else**{

clip2 = AudioSystem.*getClip*();

clip2.open(as);

clip2.start();

}

}

|  |  |
| --- | --- |
| **수행프로젝트** | |
| **작품명** | **Android Defence Game** |
| **개발기간** | **2012.10.30~12.05** |
| **개발환경** | **Java JDK 1.7.0, Windows7, Eclipse IDE, illustrator cs4** |

**아래의 내용은 Android Game Programing 프로젝트를 위해 만든 액션 디펜스 게임의 소스입니다. 미완성의 작품이지만 하나의 액티비티 안에서 터치를 통해 화면을 움직이고, 스레드를 통해 몬스터가 소환되어 같은 진행방향으로 움직이는 방식입니다.**

**결과 화면:**

****

**소스 설명:**

**게임 시작 시에 액티비티에서 화면 설정을 하고 ‘게임 시작’ 버튼을 위치시킵니다. 버튼을 클릭할 경우 main.xml이 로딩되며 GameRun클래스가 실행됩니다.**

**publicclass** DeTest2Activity **extends** Activity {

**publicvoid** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

getWindow().setFlags(WindowManager.LayoutParams.*FLAG\_FULLSCREEN*, WindowManager.LayoutParams.*FLAG\_FULLSCREEN*);

requestWindowFeature(Window.*FEATURE\_NO\_TITLE*);

setContentView(R.layout.*intro*);

Button btnStart=(Button)findViewById(R.id.*button1*);

btnStart.setOnClickListener(**new** View.OnClickListener(){

**publicvoid** onClick(View view) {

setContentView(R.layout.*main*);

} (중략)

**publicclass** GameRun **extends** SurfaceView **implements** Callback{

**private** GameThread gameThread;

**publicstaticint***touchX*, *touchY*, *touchAction*;

**public** GameRun(Context context, AttributeSet attrs) {

**super**(context, attrs);

*touchAction*=1;

SurfaceHolder holder=getHolder();

holder.addCallback(**this**);

gameThread=**new** GameThread(holder, context);

}

**publicvoid** surfaceCreated(SurfaceHolder holder) {

gameThread.start();

}

**publicboolean** onTouchEvent(MotionEvent event) {

*touchX*=(**int**)event.getX();

*touchY*=(**int**)event.getY();

*touchAction*=event.getAction();

**returntrue**;

}

}

**위의 두 클래스를 바탕으로 GameThread 클래스에서 게임 실행과 관련된 내용을 구현하게 됩니다. 게임 시작 시에 먼저 여러 이미지들을 읽고 난 후 run()메서드가 실행됩니다.**

bmpBG=BitmapFactory.*decodeResource*(res,R.drawable.*bg*);

bmpHero=BitmapFactory.*decodeResource*(res,R.drawable.*hero*);

bmpHeroLeft=BitmapFactory.*decodeResource*(res,R.drawable.*heroleft*);//용사가 왼쪽으로 향할 경우 왼쪽을 향한 이미지

bmpMop=BitmapFactory.*decodeResource*(res,R.drawable.*mop*);

**publicvoid** run(){

Hero=bmpHero; //히어로초기이미지는우측을향한것

**while**(gameStart==**true**){

canvas=surfaceHolder.lockCanvas(**null**); //락

canvas.drawBitmap(bmpBG, bmpX, 0, **null**);

canvas.drawBitmap(bmpBG, roundBGX, 0, **null**);

canvas.drawBitmap(Hero, heroX, 250, **null**);

canvas.drawBitmap(bmpMop, mopCreate.mopX, 250, **null**);

**for**(**int** i=0; i<=n; i++) {//여기서 20마리연달아생성

canvas.drawBitmap(bmpMop, (mopMgr.mop[i].mopX)-bg\_locX1, 280+(i/4\*10), **null**);

mopMgr.mop[i].mopX-=10;

**if**(mopMgr.mop[i].mopX-hero\_locX<120 &&mopMgr.mop[i].mopX-hero\_locX>=90){//몹이 용사와 마주쳤을 경우에는 지나치지 않고 용사와 마주보도록.

mopMgr.mop[i].mopX+=10;

}

**if**(mopMgr.mop[i].mopX<=50){//용사의 집 앞에 고정

mopMgr.mop[i].mopX=50;

}

}

**if**(k<=380) {//0.2초마다 thread가 돌아가므로 4초에 한번씩 몹을 소환하기 위해 변수k를 0부터 증가시켜 n값에 대입한다.

k++;

}n=(k/20);

**터치 입력을 받는 코드**

**if**(gameRun.touchAction==MotionEvent.*ACTION\_DOWN* || gameRun.touchAction==MotionEvent.*ACTION\_MOVE*){

**if**(gameRun.touchX<(w/2-50)) {//왼쪽을터치하면

Hero=bmpHeroLeft; //좌향좌용사그림

**if**(heroX<=350 &&roundBGX<=780){

bmpX+=20;

roundBGX+=20;

hero\_locX-=20;

bg\_locX1-=20; bg\_locX2-=20;

mopCreate.mopX+=20;

}

**else**{

**if**(hero\_locX>0) {

heroX-=20;

hero\_locX-=20;

(중략)

}

**elseif**(gameRun.touchX>(w/2+50)) {//오른쪽을터치하면

Hero=bmpHero; //우향후용사그림

**if**(heroX>=350 &&roundBGX>=20{

bmpX-=20

roundBGX-=20;

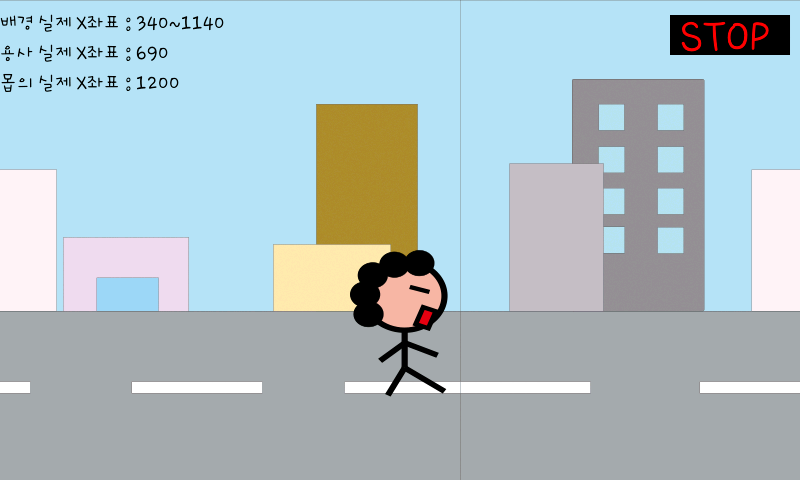
hero\_locX+=20;

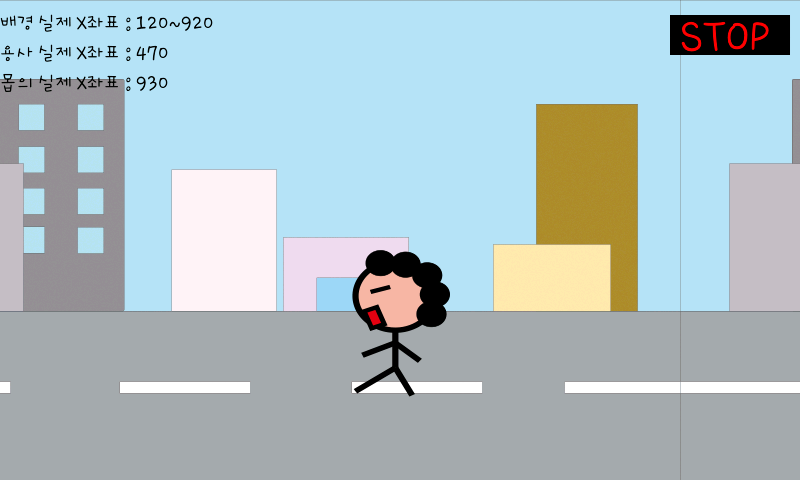
bg\_locX1+=20; bg\_locX2+=20;

mopCreate.mopX-=20;

} (중략)

**위 내용을 적용시키면 용사가 터치를 입력 받아 좌우로 움직입니다.**

****

****

**Thread가 진행되며 몬스터 소환될 때에는 몬스터와 관련된 정보를 갖고있는 MonsterCreate클래스에서 정보를 가져옵니다.**

**class** MonsterCreate //몬스터하나를생성하는클래스

{

**publicint**mopX; //몹이스크린에나타날좌표

**publicint**mop\_locX; //전체맵에서의실제몹위치좌표

**publicboolean**mop\_scope; //스크린에몹이보이는위치인가

**publicboolean**used; //몹이 활성화되었는가

**publicint**mop\_HP;

**public** MonsterCreate()

{

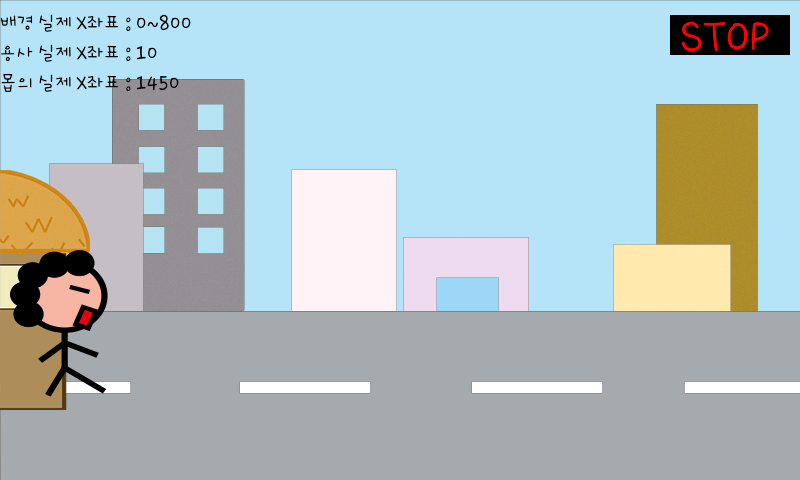
mopX=1500; //화면의제일끝에서생성. 스크린상에서움직이는거리표시

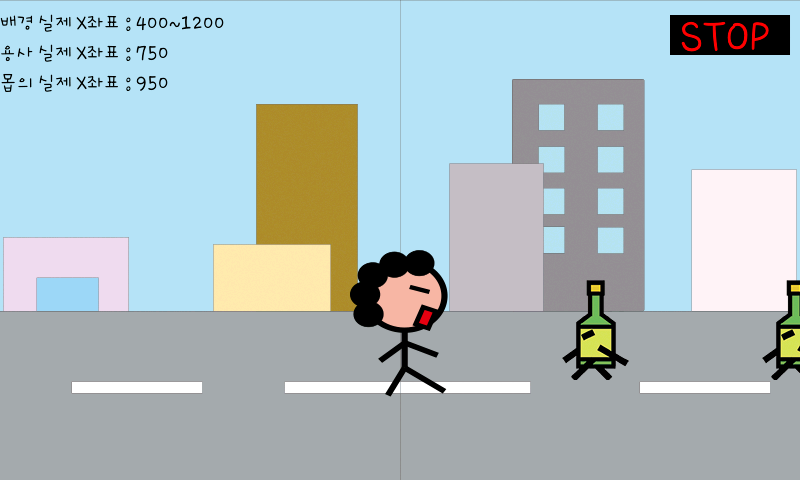
mop\_locX=1500; //실제맵에서움직이는거리표시

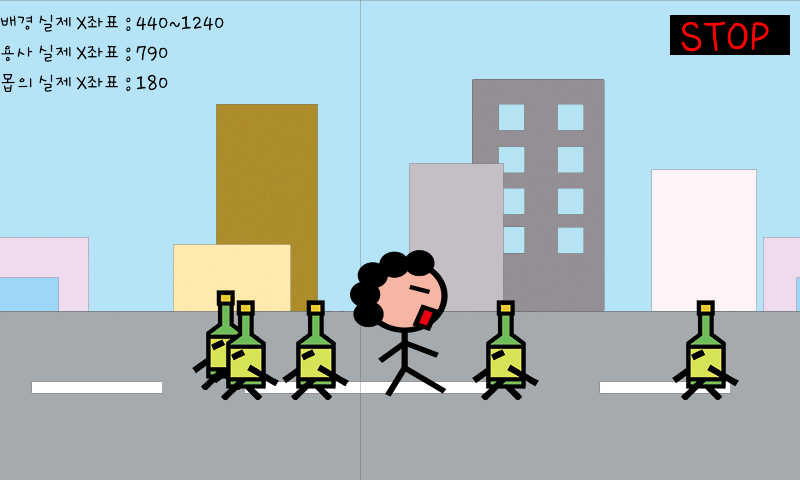
mop\_HP=100;

}

}



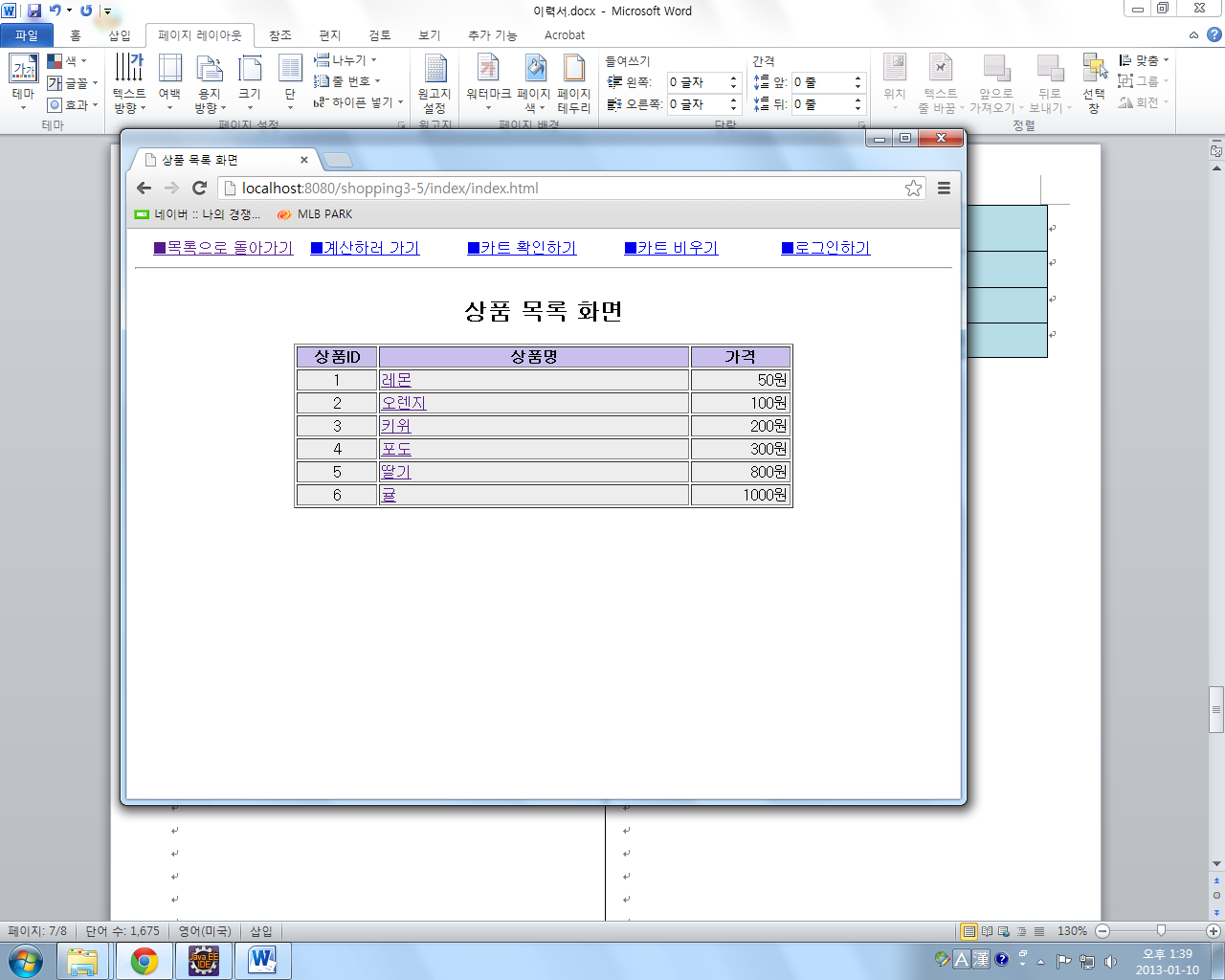




|  |  |
| --- | --- |
| **수행프로젝트** | |
| **작품명** | **Web Application Shopping Mall** |
| **개발기간** | **2012.12.10~2013.01.10** |
| **개발환경** | **Java JDK 1.7.0, Windows7, Eclipse IDE, Mysql Server 5.5, Spring Web MVC, Apache tomcat 7.0.27** |

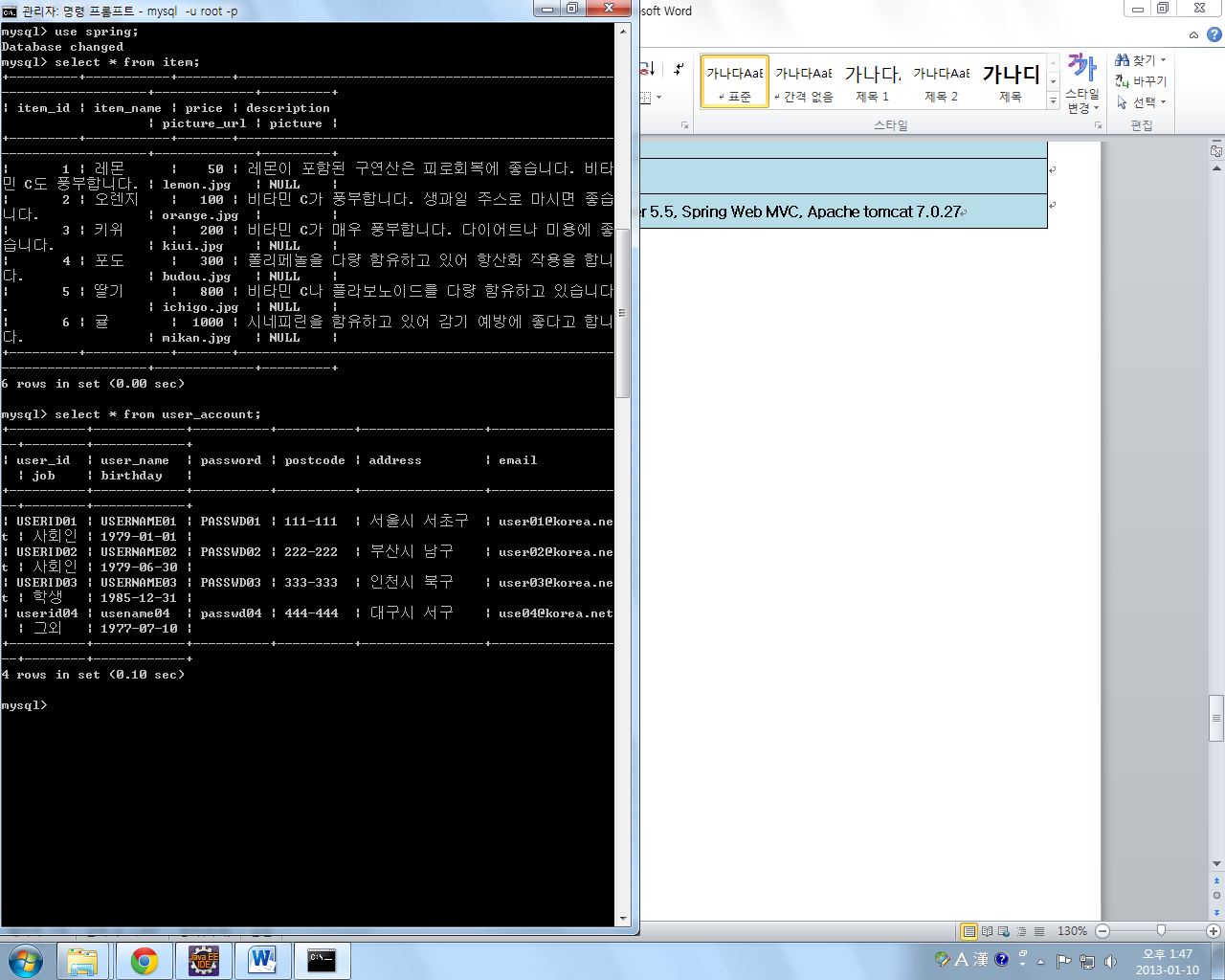
**스프링 웹 MVC와 데이터베이스를 사용한 웹 쇼핑몰 개발 프로젝트입니다. 데이터베이스로부터 상품 정보를 취득해서 웹 상에 화면을 표시하기 위해 뷰 계층인 JSP, 프리젠테이션 계층인 Spring MVC, 비즈니스 로직층, 스프링 JDBC로 분리하여 구현하였습니다.**

**결과 화면 :**



**소스 설명 :**

**MYSQL에 데이터베이스를 생성하고 필요한 테이블을 만들어 상품 자료를 입력한 후 스프링 MVC의 주요 구성 요소인 DispatcherServlet, HandlerMapping, Controller, ViewResolver, View 등을 구현합니다.**



**Web.xml**

<listener>

<listener-class>

org.springframework.web.context.ContextLoaderListener

</listener-class>

</listener>

<servlet>

<servlet-name>shopping3-5</servlet-name>

<servlet-class>

org.springframework.web.servlet.DispatcherServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>shopping3-5</servlet-name>

<url-pattern>\*.html</url-pattern>

</servlet-mapping>(중략)

**리스너로 등록된 ContextLoaderListener는 ServletContext 인스턴스 생성 시(톰캣이 애플리케이션을 읽을 때) 호출되는 것으로, DispatcherServlet 클래스가 읽기 전 ContextLoaderListener 클래스가 비즈니스 로직 계층용 설정 파일 ‘applicationContext.xml’ 을 읽습니다.**

**applicationContext.xml**

<!-- Data Source -->

<beanid=*"dataSource"*class=*"org.springframework.jdbc.datasource.DriverManagerDataSource"*>

<propertyname=*"driverClassName"*>

<value>com.mysql.jdbc.Driver</value></property>

<propertyname=*"url"*>

<value>jdbc:mysql://localhost/spring?useUnicode=true&amp;characterEncoding=utf8</value></property>

(중략)

<!-- Transaction -->

<beanid=*"transactionManager"*class=*"org.springframework.jdbc.datasource.DataSourceTransactionManager"* p:dataSource-ref=*"dataSource"*></bean>

<aop:config>

<aop:pointcutid=*"serviceMethods"*expression=*"execution(\* logic.ShopImpl.\*(..))"*/>

<aop:advisoradvice-ref=*"txAdvice"*pointcut-ref=*"serviceMethods"*/></aop:config>

<tx:adviceid=*"txAdvice"*transaction-manager=*"transactionManager"*>

<tx:attributes>

<tx:methodname=*"entryUser"*propagation=*"REQUIRED"*rollback-for=*"Exception"*/>

<tx:methodname=*"checkout"*propagation=*"REQUIRED"*rollback-for=*"Exception"*/>

<tx:methodname=*"\*"*propagation=*"SUPPORTS"*read-only=*"true"*/></tx:attributes></tx:advice>

(중략)

**DataSource에서는 데이터베이스에 접속하기 위한 데이터소스를 정의하고 있으며 Transaction은 비즈니스 로직층에 적용됩니다. 이것으로 인해 예외가 발생할 경우에는 어떤 예외라도 롤백하도록 rollback-for 속성을 지정하였습니다. 그리고 tx:method 속성을 통해 메서드별로 적용을 달리 하였습니다.**

**Shoppinrg3-5-servlet.xml**

<!-- HandlerMapping -->

<beanclass=*"org.springframework.web.servlet.mvc.support.ControllerClassNameHandlerMapping"*/>

<context:component-scanbase-package=*"controller,dao,logic"*/>

<!-- ViewResolver -->

<beanid=*"internalResourceViewResolver"*class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

<propertyname=*"viewClass"*>

<value>org.springframework.web.servlet.view.JstlView</value>

</property>

<propertyname=*"prefix"*><value>/WEB-INF/jsp/</value>

</property>

<propertyname=*"suffix"*><value>.jsp</value>

</property>

</bean>

<!-- ExceptionHandler -->

<beanid=*"exceptionHandler"*

class=*"org.springframework.web.servlet.handler.SimpleMappingExceptionResolver"*>

<propertyname=*"exceptionMappings"*>

(중략)

**HandlerMapping 에서 ControllerClassNameHandlerMapping 클래스는 컨트롤러의 클래스 이름에서 URL을 추출하여 해당 페이지로 이동합니다. Context:component-scan 에서 클래스를 탐색할 패키지를 지정하였습니다. ViewResolver에서는 property에 지정된 경로에서 해당 페이지를 탐색하여 화면에 표시합니다. 그리고 ExceptionHandler 는 애플리케이션에서 예외가 발생한 경우 각각의 오류 페이지로 이동하게 합니다.**

**IndexController.java**

@Controller

**publicclass** IndexController {

@Autowired

**private** Shop shopService;

@RequestMapping

**public** ModelAndView index(HttpSession session) {

// 상품목록정보를취득

List<Item> itemList = **this**.shopService.getItemList();

// 모델생성

Map<String, Object> model = **new** HashMap<String, Object>();

model.put("itemList", itemList);

// 반환값이될 ModelAndView 인스턴스생성

ModelAndView modelAndView = **new** ModelAndView();

modelAndView.addAllObjects(model);

User loginUser= (User)session.getAttribute(WebConstants.*USER\_KEY*);

**if** (loginUser != **null**) {

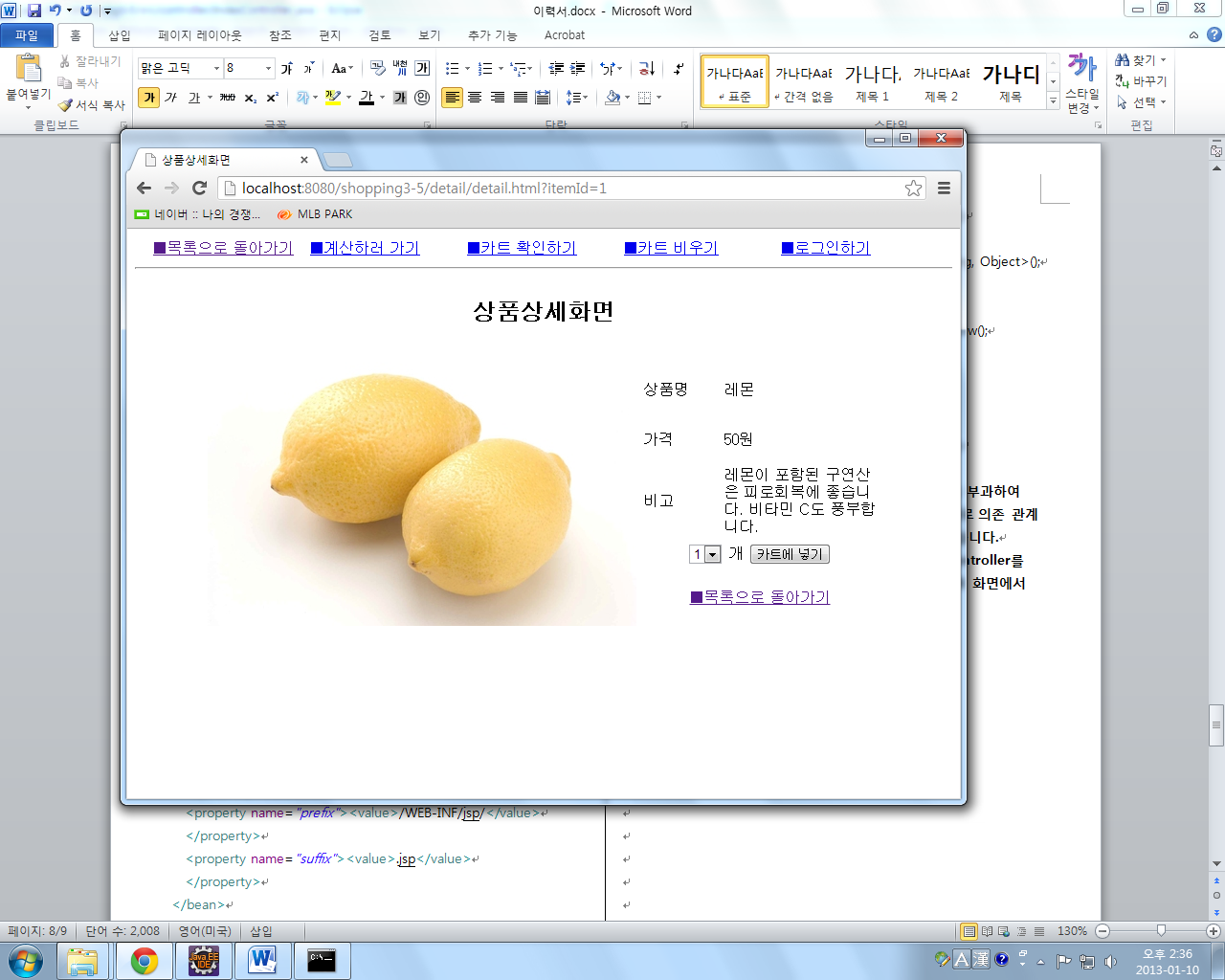
modelAndView.addObject("loginUser", loginUser);

}**return** modelAndView;

(중략)

**Contoller 인터페이스를 구현하지 않고 어노테이션을 부과하여 컨트롤러 클래스를 구현하였습니다. 또한 @Autowired로 의존 관계 주입을 자동화하여 설정 메서드를 따로 정의하지 않았습니다.**

**IndexController와 비슷한 형식으로 작성된 DetailController를 통해 상품의 상세화면을 확인할 수 있습니다. 상품 목록 화면에서 상품명을 클릭하면 해당 페이지로 이동합니다.**



<%@includefile=*"/WEB-INF/jsp/cart\_header.jsp"*%>

<divalign=*"center"*class=*"body"*>

<h2>상품상세화면</h2><table><tr>

<td><imgsrc=*"../img/*${item.pictureUrl}*"*></td>

<tdalign=*"center"*>

<table>

<trheight=*"50"*>

<tdwidth=*"80"*>상품명</td>

<tdwidth=*"160"*>${item.itemName}</td></tr>

<trheight=*"50"*>

<tdwidth=*"80"*>가격</td>

<tdwidth=*"160"*>${item.price}원</td></tr>

<trheight=*"50"*>

<tdwidth=*"80"*>비고</td>

<tdwidth=*"160"*>${item.description}</td></tr>

<tr>

<tdcolspan=*"2"*align=*"center"*width=*"230"*>

<formaction=*"../cart/cartAdd.html"*>

<inputtype=*"hidden"*name=*"itemId"*value=*"*${item.itemId}*"*>

<table><tr><td><selectname=*"quantity"*>

<option>1</option><option>2</option><option>3</option><option>4</option><option>5</option><option>6</option><option>7</option><option>8</option><option>9</option>

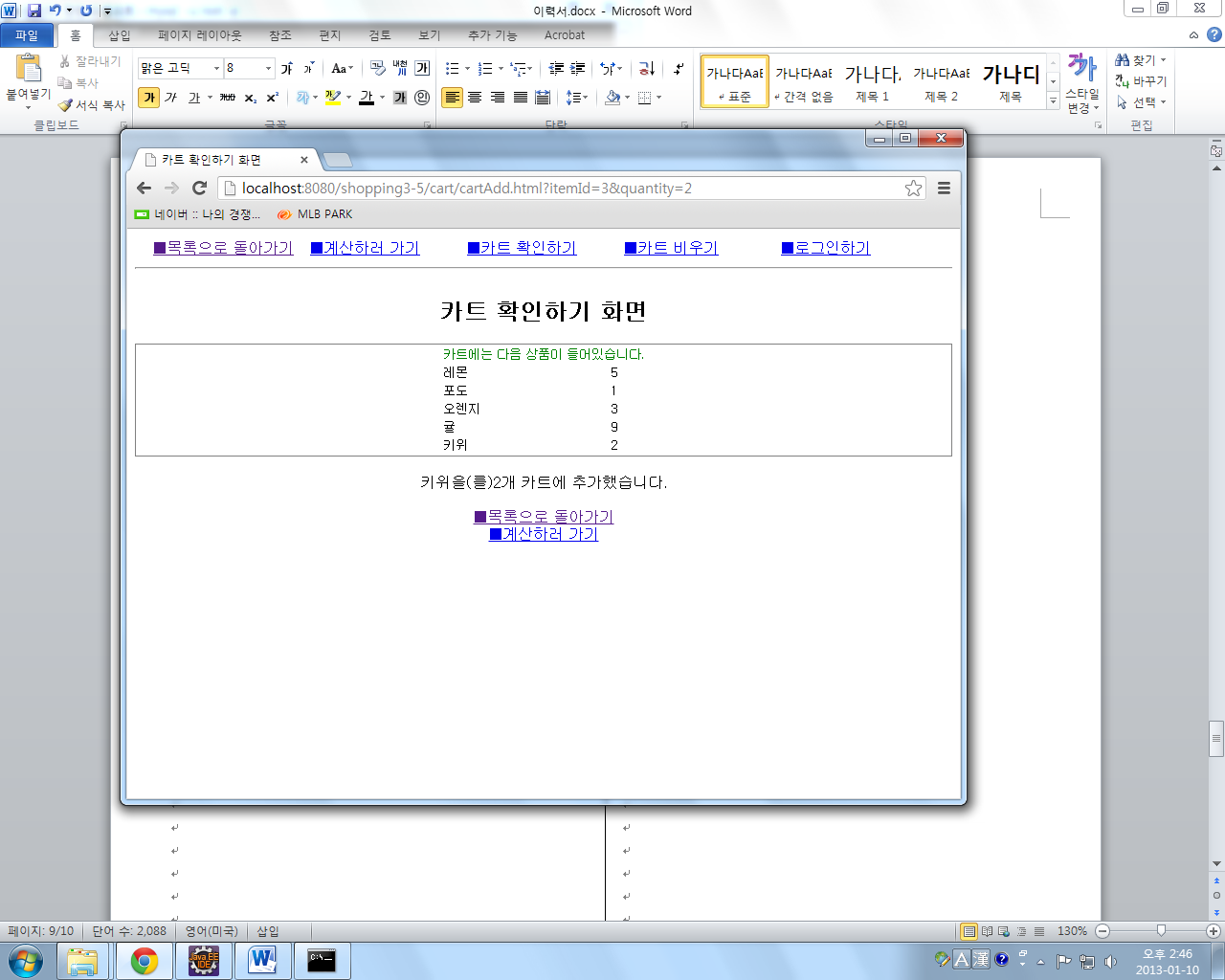
</select>&nbsp;개</td>

<td><inputtype=*"submit"*value=*"카트에넣기"*></td>

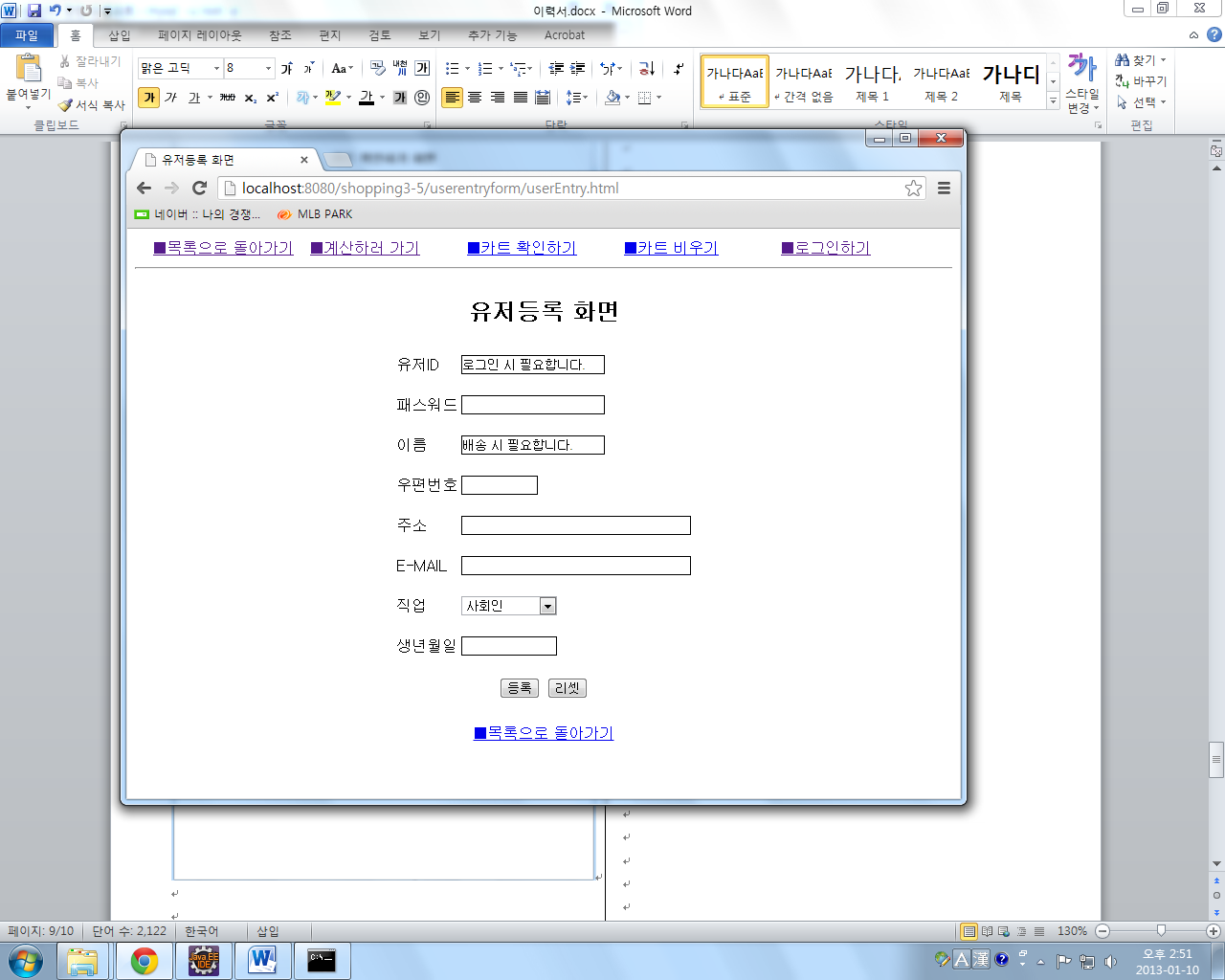
</tr>

(중략)

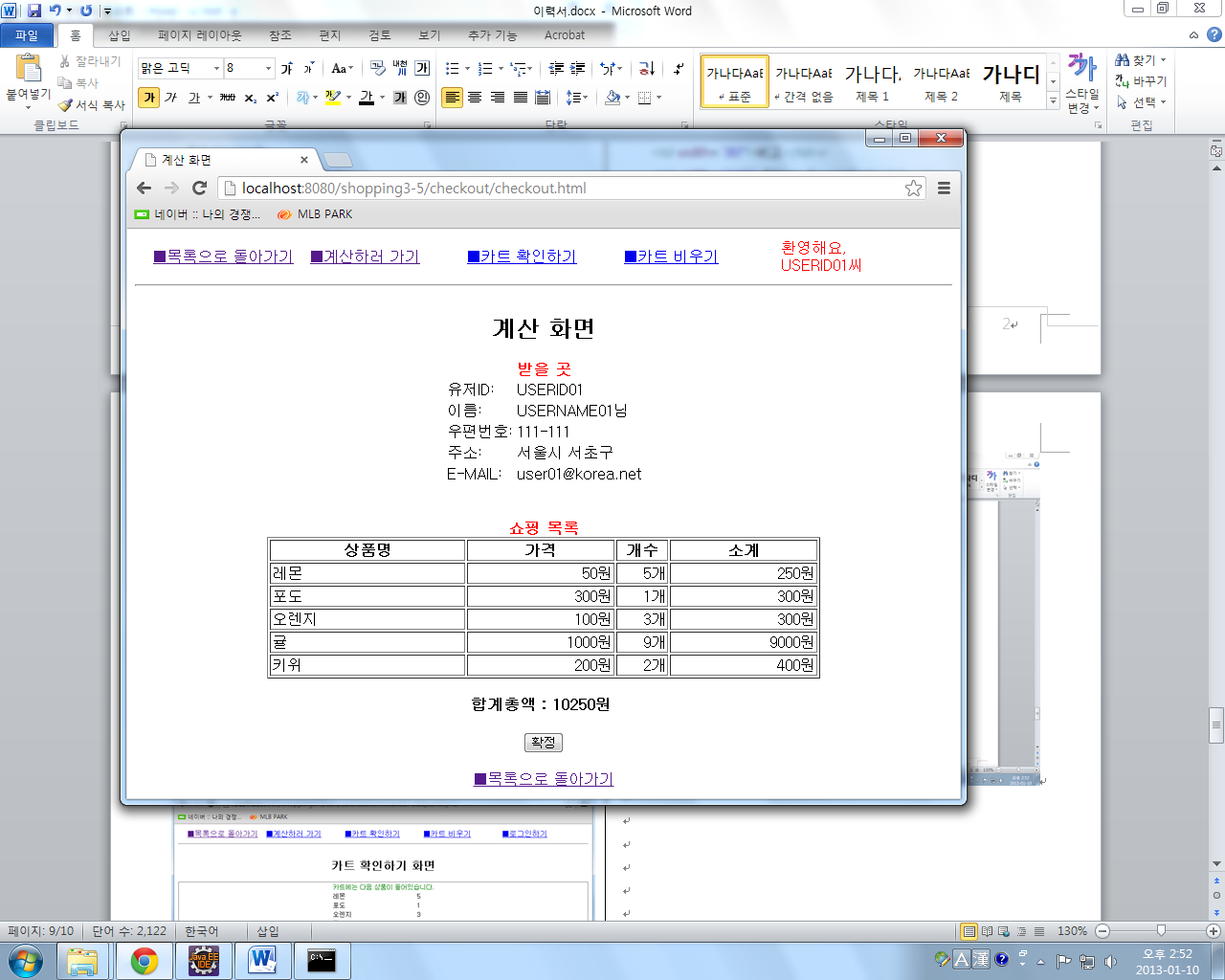
**상품 상세 화면을 표시하는 detail.jsp 에구입하는 수량을 지정하는 선택 목록과 ‘카트에 넣기’버튼을 추가하였습니다. ‘카트에 넣기’버튼을 클릭하면, 상품 ID를 요청 파라미터로 보내, 카트에 해당하는 상품을 추가합니다. 그리고 카트 확인하기 화면으로 이동합니다.**



**그리고 ‘계산하러 가기’를 클릭하면 로그인 화면으로 이동합니다. 미리 만들었던 데이터베이스에 user\_account 테이블에서 유저 목록을 관리하며 아이디나 패스워드가 바르게 입력되지 않은 경우에는 예외가 발생하여 다시 로그인 화면으로 이동합니다. 유저 등록이 되어있지 않을 경우 신규유저등록이 가능합니다.**



**로그인이 완료되면 화면 우측 상단의 ‘로그인하기’폼이 ‘환영해요 XX님’ 이라는 내용으로 바뀌며 계산창으로 이동이 가능합니다.**



**상품, 유저, 카트에 관한 정보들은 비즈니스 로직으로서 도메인 객체 목록으로 따로 관리합니다. 여기에서 사용된 도메인 객체는 Shop, UserCatalog, User, ItemCatalog, Item, SaleCatalog, Sale, SaleLine, Cart, ItemSet 등이 있으며 이 도메인 객체들이 협조해서 동작하면서 비즈니스 로직을 구성합니다.**

@Repository

**publicclass** ItemDaoImpl **implements** ItemDao {

**private** SimpleJdbcTemplate template;

@Autowired

**publicvoid** setDataSource(DataSource dataSource) {

**this**.template = **new** SimpleJdbcTemplate(dataSource);

}

**privatestaticfinal** String *SELECT\_ALL* = "SELECT item\_id, item\_name, price, description, picture\_url FROM item";

**public** List<Item> findAll() {

RowMapper<Item> mapper = **new** BeanPropertyRowMapper<Item>(Item.**class**);

**returnthis**.template.query(ItemDaoImpl.*SELECT\_ALL*, mapper);

}

**privatestaticfinal** String *SELECT\_BY\_PRIMARY\_KEY* = "SELECT item\_id, item\_name, price, description, picture\_url FROM item WHERE item\_id = ?";

**public** Item findByPrimaryKey(Integer itemId) {

RowMapper<Item> mapper = **new** BeanPropertyRowMapper<Item>(Item.**class**);

**returnthis**.template.queryForObject(*SELECT\_BY\_PRIMARY\_KEY*, mapper, itemId);

}

}

**SimpleJdbcTemplate클래스를 이용해서 데이터베이스에 쿼리를 발행합니다. ‘?’기호를 써서 쿼리를 작성할 수 있는데 쿼리에 지정할 실제 값은 인수로 넘깁니다.**

**이처럼 도메인 모델이 표현하는 비즈니스 규칙에서 데이터 접근 처리를 분리하기 위해 DAO계층을 사용하였습니다. 여기에서는 ItemDao, UserDao, SaleDao, SaleLineDao등이 사용되었습니다.**