package intern;  
  
import appt.meta3.\*;  
import appt.meta3.servlet.AuthServlet;  
import org.apache.commons.lang3.math.NumberUtils;  
  
import javax.servlet.http.Cookie;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
import java.io.PrintWriter;  
import java.nio.charset.StandardCharsets;  
import java.text.\*;  
import java.util.\*;  
import java.util.function.BiConsumer;  
  
public class EldarServlet extends HttpServlet {  
 private PrintWriter out;  
 private ResourceBundle mains;  
 private HttpServletRequest request;  
 private HttpServletResponse response;  
 private final int myId = 1000360;  
 private final String redis = "rev";  
 private final String prefix = "Eldar";  
  
 private final Map<String, String> typeMap = new HashMap<>();  
 private final Map<String, String> yesnoMap = new HashMap<>();  
 private final Map<String, List<String>> pagesMap = new TreeMap<>();  
 private List<Procedure> solutionList;  
  
 public interface Procedure {  
 void run() throws Exception;  
 }  
  
 public void doPost(HttpServletRequest req, HttpServletResponse res) throws IOException {  
 initialize(req, res);  
 String task = request.getParameter("task");  
 try {  
 if ("7".equals(task)) {  
 lection3task7post();  
 }  
 if ("13".equals(task)) {  
 lection5task1post();  
 }  
 } catch (Exception e) {  
 printException(e);  
 }  
 out.flush();  
 }  
  
 public void doGet(HttpServletRequest req, HttpServletResponse res) throws IOException {  
 initialize(req, res);  
 try {  
 int task = Integer.*parseInt*(request.getParameter("task"));  
 if(task < 1 || task > solutionList.size()) {  
 throw new NumberFormatException();  
 }  
 solutionList.get(task - 1).run();  
 } catch (NumberFormatException e) {  
 printHtml("Hello", "<h1 style=\"text-align:center;\">Привет Sirius!</h1>");  
 } catch (Exception e) {  
 printException(e);  
 }  
 out.flush();  
 }  
  
 public void initialize(HttpServletRequest req, HttpServletResponse res) throws IOException {  
 request = req;  
 response = res;  
 response.setCharacterEncoding("UTF-8");  
 request.setCharacterEncoding("UTF-8");  
 out = res.getWriter();  
 mains = ResourceBundle.*getBundle*("intern\_main", new Locale("ru", "RU"));  
  
 typeMap.put("", "-");  
 typeMap.put("0", "Экскурсия");  
 typeMap.put("1", "Билет");  
 typeMap.put("2", "Спорт");  
 typeMap.put("3", "Прокат");  
 typeMap.put("4", "Услуга");  
 typeMap.put("5", "СПА");  
 typeMap.put("6", "Авиация");  
 typeMap.put("8", "Концерт");  
  
 yesnoMap.put("", "-");  
 yesnoMap.put("1", "Да");  
 yesnoMap.put("0", "Нет");  
  
 pagesMap.put("PostgreSQL", Arrays.*asList*("Один", "Два",  
 "Три", "Четыре", "Пять", "Шесть", "Семь", "Восемь"));  
 pagesMap.put("Redis", Arrays.*asList*("Один", "Два", "Три", "Четыре"));  
 pagesMap.put("Авторизация", Arrays.*asList*("Один", "Два", "Три", "Четыре", "Пять"));  
  
 solutionList = Arrays.*asList*(this::lection3task1, this::lection3task2, this::lection3task3, this::lection3task4,  
 this::lection3task5, this::lection3task6, this::lection3task7, this::lection3task8, this::lection4task1,  
 this::lection4task2, this::lection4task3, this::lection4task4, this::lection5task1, this::lection5task2,  
 this::lection5task3, this::lection5task4, this::lection5task5, this::cheatCode, this::cheatFunction);  
 }  
  
 public void printException(Exception e) {  
 StackTraceElement[] stack = e.getStackTrace();  
 String[] stackTrace = new String[stack.length];  
 for(int i = 0; i < stack.length; ++i) {  
 stackTrace[i] = stack[i].toString();  
 }  
 printHtml("Exception!", String.*format*("<h1>%s: %s</h1><div>%s</div>",  
 e.getClass().getName(), e.getMessage(), String.*join*("<br>", stackTrace)));  
 }  
  
 public void lection3task1() throws Exception {  
 Map<String, String> regions = getRegions("100410000050");  
 Obb filter = Ob0.*createFilter*(5);  
 String regionId = request.getParameter("region");  
 if(regionId == null || regionId.equals("")) {  
 regionId = "100518301512";  
 }  
 Ob0.*addCondition*(filter, 1005101368, Ob0.ComparisonType.*EQ*, regionId);  
 Obb[] cities = Ob0.*getSrcObs*(mains, filter, 0, 0);  
 List<String> regionIds = intern.Utils.*getKeysSortedByValue*(regions, false);  
 StringBuilder text = new StringBuilder("<form action=\"#\" method=\"get\">" +  
 "<table class=\"form\"><tr><td>Регион</td><td><select name=\"region\">");  
 for(String key : regionIds) {  
 text.append(String.*format*("<option%s value=\"%s\">%s</option>",  
 key.equals(regionId) ? " selected" : "", key, regions.get(key)));  
 }  
 text.append("</select></td></tr></table><input type=\"submit\" value=\"Выбрать\">" +  
 "<input type=\"hidden\" name=\"task\" value=\"1\"></form>");  
 Arrays.*sort*(cities, Comparator.*comparing*((Obb ob) -> ob.getAt("1000098")));  
 int i = 0;  
 text.append("<table class=\"data\"><tr><td>№</td><td>ID</td><td>Город</td></tr>");  
 for(Obb city : cities) {  
 text.append(String.*format*("<tr><td>%d</td><td>%s</td><td>%s</td></tr>",  
 ++i, city.id, Ob0.*getAt*(city, "1000098")));  
 }  
 text.append("</table>");  
 printHtml("Города России", text.toString());  
 }  
  
 public Map<String, String> getRegions(String countryId) throws Exception {  
 Obb filter = Ob0.*createFilter*(5);  
 Ob0.*addCondition*(filter, 1000004, Ob0.ComparisonType.*EQ*, countryId);  
 Ob0.*addCondition*(filter, 1005101368, Ob0.ComparisonType.*NEQ*, "");  
 Obb[] cities = Ob0.*getSrcObs*(mains, filter, 0, 0);  
 Map<String, String> regions = new TreeMap<>();  
 for(Obb city : cities) {  
 String regionId = Ob0.*getAt*(city, 1005101368);  
 if(!regions.containsKey(regionId)) {  
 regions.put(regionId, Ob0.*getZn*(mains, regionId, 1000098, 4));  
 }  
 }  
 return regions;  
 }  
  
 public void lection3task2() throws Exception {  
 String countryId = request.getParameter("country");  
 if(countryId == null || countryId.equals("")) {  
 countryId = "100410000050";  
 }  
 Obb filter = Ob0.*createFilter*(5);  
 Ob0.*addCondition*(filter, 1000004, Ob0.ComparisonType.*EQ*, countryId);  
 Ob0.*addCondition*(filter, 1000101, Ob0.ComparisonType.*EQ*, "Да");  
 Obb[] cities = Ob0.*getSrcObs*(mains, filter, 0, 0);  
 Obb[] countries = Ob0.*getSrcObs*(mains, Ob0.*createFilter*(4), 0, 0);  
 Map<String, String> regions = getRegions(countryId);  
 Map<String, Set<String>> distribution = regionDistribute(cities);  
 List<String> regionIds = intern.Utils.*getKeysSortedByValue*(regions, false);  
 Arrays.*sort*(countries, Comparator.*comparing*((Obb ob) -> ob.getAt("1000000")));  
 StringBuilder text = new StringBuilder();  
 text.append("<form action=\"#\" method=\"get\"><table class=\"form\"><tr><td>" +  
 "Страна</td><td><select name=\"country\">");  
 for(Obb country : countries) {  
 text.append(String.*format*("<option%s value=\"%s\">%s</option>",  
 country.id.equals(countryId) ? " selected" : "",  
 country.id, country.getAt("1000000")));  
 }  
 text.append("</select></td></tr></table><input type=\"submit\" value=\"Выбрать\">" +  
 "<input type=\"hidden\" name=\"task\" value=\"2\"></form>");  
 if(regionIds.isEmpty()) {  
 text.append("<h4 style=\"text-align:center;color:grey;\">" +  
 "Данная страна не поддерживает разбиение на регионы</h4>");  
 } else {  
 text.append("<table class=\"data\"><tr><td>№</td><td>Область</td><td>Города</td></tr>");  
 int i = 0;  
 for (String key : regionIds) {  
 if (regions.containsKey(key) && distribution.containsKey(key)) {  
 text.append(String.*format*("<tr><td>%d</td><td>%s</td><td>%s</td></tr>",  
 ++i, regions.get(key),  
 String.*join*(", ", distribution.get(key))));  
 }  
 }  
 text.append("</table>");  
 }  
 printHtml("Города России", text.toString());  
 }  
  
 public Map<String, Set<String>> regionDistribute(Obb[] cities) {  
 Map<String, Set<String>> distribution = new TreeMap<>();  
 for(Obb city : cities) {  
 String regionId = Ob0.*getAt*(city, 1005101368);  
 if(!distribution.containsKey(regionId)) {  
 distribution.put(regionId, new TreeSet<>());  
 }  
 distribution.get(regionId).add(Ob0.*getAt*(city, 1000098));  
 }  
 return distribution;  
 }  
  
 public void lection3task3() throws Exception {  
 response.setContentType("application/json; charset=UTF-8");  
 String birthday = request.getParameter("bd");  
 if(birthday == null || birthday.equals("")) {  
 birthday = "01.01.1990";  
 }  
 Obb filter = Ob0.*createFilter*(23);  
 Ob0.*addCondition*(filter, 1000152, Ob0.ComparisonType.*GT*, birthday);  
 Obb[] tourists = Ob0.*getSrcObs*(mains, filter, 0, 0);  
 Arrays.*sort*(tourists, Comparator.*comparing*(  
 (Obb ob) -> ob == null ? "" : ob.getAt("1000144")));  
  
 out.print("[");  
 int i = 0;  
 for(Obb tourist : tourists) {  
 out.printf("{\"id\":\"%s\",\"nm\":\"%s %s %s\",\"bd\":\"%s\",\"trs\":[",  
 tourist == null ? "null" : tourist.id,  
 Ob0.*getAt*(tourist, 1000144),  
 Ob0.*getAt*(tourist, 1000146),  
 Ob0.*getAt*(tourist, 1000147),  
 Ob0.*getAt*(tourist, 1000152));  
 String[] tours = Ob0.*getAt*(tourist, 1023422081).split("#");  
 int j = 0;  
 for(String tour : tours) {  
 out.printf("\"%s\"%s", tour, ++j < tours.length ? "," : "");  
 }  
 out.printf("]}%s", ++i < tourists.length ? "," : "");  
 }  
 out.print("]");  
 }  
  
 public void lection3task4() throws Exception {  
 String agentName = request.getParameter("agent");  
 String categoryName = request.getParameter("category");  
 if(agentName == null || agentName.equals("")) {  
 agentName = "КонтрАг";  
 }  
 if(categoryName == null || categoryName.equals("")) {  
 categoryName = "Концерты";  
 }  
 Obb filter = Ob0.*createFilter*(36);  
 Ob0.*addCondition*(filter, new int[]{1036922797, 1317100000},  
 Ob0.ComparisonType.*EQ*, agentName);  
 Ob0.*addCondition*(filter, new int[]{1036900082, 1162100000},  
 Ob0.ComparisonType.*EQ*, categoryName);  
 Obb[] costs = Ob0.*getSrcObs*(mains, filter, 0, 0);  
 Obb[] agents = Ob0.*getSrcObs*(mains, Ob0.*createFilter*(317), 0, 0);  
 Obb[] categories = Ob0.*getSrcObs*(mains, Ob0.*createFilter*(162), 0, 0);  
 Arrays.*sort*(costs, (Obb left, Obb right) -> right.data\_n.compareTo(left.data\_n));  
 costs = Arrays.*copyOfRange*(costs, Math.*max*(0, costs.length - 50), costs.length);  
 Arrays.*sort*(costs, Comparator.*comparing*((Obb ob) -> ob.getAt("1036423021")));  
 Arrays.*sort*(agents, Comparator.*comparing*((Obb ob) -> ob.getAt("1317100000")));  
 StringBuilder text = new StringBuilder();  
 text.append("<form action=\"#\" method=\"get\"><table class=\"form\"><tr><td>" +  
 "Контрагент</td><td><select name=\"agent\">");  
 for(Obb agent : agents) {  
 String an = agent.getAt("1317100000");  
 text.append(String.*format*("<option%s>%s</option>",  
 an.equals(agentName) ? " selected" : "", an));  
 }  
 text.append("</select></td></tr><tr><td>Категория</td><td><select name=\"category\">");  
 for(Obb category : categories) {  
 String cn = category.getAt("1162100000");  
 text.append(String.*format*("<option%s>%s</option>",  
 cn.equals(categoryName) ? " selected" : "", cn));  
 }  
 text.append("</select></td><tr></table><input type=\"submit\" value=\"Выбрать\">" +  
 "<input type=\"hidden\" name=\"task\" value=\"4\"></form>" +  
 "<table class=\"data\"><tr><td>№</td><td>Название</td>" +  
 "<td>Категория</td><td>Адрес</td></tr>");  
 int i = 0;  
 for(Obb cost : costs) {  
 text.append(String.*format*("<tr><td>%d</td><td>%s</td><td>%s</td><td>%s</td></tr>",  
 ++i, cost.getAt("1036423021"),  
 yesnoMap.get(cost.getAt("1036200042")),  
 cost.getAt("1036410028")));  
 }  
 printHtml("Экскурсии", text.append("</table>").toString());  
 }  
  
 public void lection3task5() throws Exception {  
 Obb[] costs = Ob0.*getSrcObs*(mains, Ob0.*createFilter*(36), 0, 0);  
 List<Obb> list = Arrays.*asList*(costs);  
 List<Obb> array = new ArrayList<>(list);  
 List<Obb> linked = new LinkedList<>(list);  
 StringBuilder text = new StringBuilder();  
 BiConsumer<List<Obb>, String> consumer = (List<Obb> lst, String listType) -> {  
 int i = 0;  
 text.append("<div>");  
 long time = System.*currentTimeMillis*();  
 for(Obb item : lst) {  
 text.append(String.*format*("[%d : %s]%s", ++i, item.id, i == lst.size() ? "" : ", "));  
 }  
 time = System.*currentTimeMillis*() - time;  
 text.append(String.*format*("<br>%s - %d ms</div>", listType, time));  
 };  
 consumer.accept(array, "ArrayList");  
 consumer.accept(linked, "LinkedList");  
 printHtml("Временное сравнение", text.toString());  
 }  
  
 public void lection3task6() throws Exception {  
 String type = request.getParameter("type");  
 String code = request.getParameter("code");  
 String address = request.getParameter("address");  
 String category = request.getParameter("category");  
 String agentId = request.getParameter("agent");  
  
 type = NumberUtils.*isNumber*(type) && Integer.*parseInt*(type) >= 0  
 && Integer.*parseInt*(type) <= 8 && Integer.*parseInt*(type) != 7 ? type : "";  
 code = code == null ? "" : code;  
 address = address == null ? "" : address;  
 category = category == null || !(category.equals("1") || category.equals("0")) ? "" : category;  
 agentId = agentId == null ? "" : agentId;  
  
 Obb filter = Ob0.*createFilter*(36);  
 if(!type.equals("")) {  
 Ob0.*addCondition*(filter, 1036200042, Ob0.ComparisonType.*EQ*, type);  
 }  
 if(!code.equals("")) {  
 Ob0.*addCondition*(filter, 1036423021, Ob0.ComparisonType.*EQ*, code);  
 }  
 if(!address.equals("")) {  
 Ob0.*addCondition*(filter, 1036410028, Ob0.ComparisonType.*EQ*, address);  
 }  
 if(!category.equals("")) {  
 Ob0.*addCondition*(filter, 1162200125, Ob0.ComparisonType.*EQ*, category);  
 }  
 if(!agentId.equals("")) {  
 Ob0.*addCondition*(filter, 1036922797, Ob0.ComparisonType.*EQ*, agentId);  
 }  
 Obb[] costs = Ob0.*getSrcObs*(mains, filter, 0, 0);  
 Obb[] agents = Ob0.*getSrcObs*(mains, Ob0.*createFilter*(317), 0, 0);  
 Arrays.*sort*(agents, Comparator.*comparing*((Obb ob) -> ob.getAt("1317100000")));  
  
 Map<String, String> agentMap = new HashMap<>();  
 agentMap.put("", "-");  
 for(Obb agent : agents) {  
 agentMap.put(agent.id, agent.getAt("1317100000"));  
 }  
  
 StringBuilder text = new StringBuilder();  
 text.append("<form action=\"#\" method=\"get\"><table class=\"form\"><tr><td>" +  
 "Тип</td><td><select name=\"type\">");  
 for(String key : typeMap.keySet()) {  
 text.append(String.*format*("<option%s value=\"%s\">%s</option>",  
 type.equals(key) ? " selected" : "", key, typeMap.get(key)));  
 }  
 text.append("</select></td></tr><tr><td>Код</td>" +  
 "<td><input name=\"code\" type=\"text\" value=\"\"></td></tr>" +  
 "<tr><td>Адрес</td><td><input name=\"address\" type=\"text\" value=\"\">" +  
 "</td></tr><tr><td>Категория</td><td><select name=\"category\">");  
 for(String key : yesnoMap.keySet()) {  
 text.append(String.*format*("<option%s value=\"%s\">%s</option>",  
 category.equals(key) ? " selected" : "", key, yesnoMap.get(key)));  
 }  
 text.append("</select></td></tr><tr><td>Контрагент</td><td><select name=\"agent\">");  
 for(String key : agentMap.keySet()) {  
 text.append(String.*format*("<option%s value=\"%s\">%s</option>",  
 agentId.equals(key) ? " selected" : "", key, agentMap.get(key)));  
 }  
 text.append("</select></td></tr></table><input type=\"submit\" value=\"Выбрать\">" +  
 "<input type=\"hidden\" name=\"task\" value=\"6\"></form>" +  
 "<table class=\"data\"><tr><td>№</td><td>Название</td><td>Тип</td>" +  
 "<td>Адрес</td><td>Да/Нет</td><td>Контрагент</td></tr>");  
 Arrays.*sort*(costs, Comparator.*comparing*((Obb ob) -> ob.getAt("1036423021")));  
 int i = 0;  
 for(Obb cost : costs) {  
 text.append(String.*format*("<tr><td>%d</td><td>%s</td><td>%s</td>" +  
 "<td>%s</td><td>%s</td><td>%s</td></tr>",  
 ++i, cost.getAt("1036423021"),  
 typeMap.get(cost.getAt("1036200042")),  
 cost.getAt("1036410028"),  
 yesnoMap.get(cost.getAt("1162200125")),  
 agentMap.get(cost.getAt("1036922797"))));  
 }  
 printHtml("Экскурсии", text.append("</table>").toString());  
 }  
  
 public void lection3task7() throws Exception {  
 Map<String, String> regions = new HashMap<>();  
 Map<String, String> partners = new HashMap<>();  
 regionsAndPartners(regions, partners);  
  
 StringBuilder text = new StringBuilder();  
 text.append("<form method=\"post\" action=\"#\"><table class=\"form\">" +  
 "<tr><td>Название</td><td><input type=\"text\" name=\"name\"></td></tr>" +  
 "<tr><td>Описание</td><td><textarea name=\"desc\"></textarea></td></tr>" +  
 "<tr><td>Регион</td><td><select name=\"region\">");  
 List<String> regionIds = intern.Utils.*getKeysSortedByValue*(regions, false);  
 for(String regionId : regionIds) {  
 text.append(String.*format*("<option value=\"%s\">%s</option>",  
 regionId, regions.get(regionId)));  
 }  
 text.append("</select></td></tr>" +  
 "<tr><td>Доп. оплата</td><td><input type=\"text\" name=\"cost\"></td></tr>" +  
 "<tr><td>Бронирование у партнера</td><td><select name=\"partner\">");  
 for(String partnerId : partners.keySet()) {  
 text.append(String.*format*("<option value=\"%s\">%s</option>",  
 partnerId, partners.get(partnerId)));  
 }  
 text.append("</select></td></tr>" +  
 "<tr><td>Тип</td><td><select name=\"type\">");  
 for(String typeId : typeMap.keySet()) {  
 text.append(String.*format*("<option value=\"%s\">%s</option>",  
 typeId, typeMap.get(typeId)));  
 }  
 text.append("</select></td></tr>" +  
 "</table><input type=\"hidden\" name=\"task\" value=\"7\">" +  
 "<input type=\"submit\" value=\"Создать\"></form>");  
 printDescs(text, regions, partners);  
 }  
  
 public void lection3task7post() throws Exception {  
 String name = request.getParameter("name");  
 String desc = request.getParameter("desc");  
 String region = request.getParameter("region");  
 String cost = request.getParameter("cost");  
 String partner = request.getParameter("partner");  
 String type = request.getParameter("type");  
  
 name = name == null ? "" : name;  
 desc = desc == null ? "" : desc;  
 region = region == null ? "-" : region;  
 cost = !NumberUtils.*isNumber*(cost) ? "" : cost;  
 partner = partner == null ? "-" : partner;  
 type = !typeMap.containsKey(type) ? "-" : type;  
  
 if(!name.equals("") && !desc.equals("") && !region.equals("-") &&  
 !cost.equals("") && !partner.equals("-") && !type.equals("-")) {  
 Obb ob = new Obb(506);  
 ob.id\_user = myId;  
 Ob0.*addAt*(ob, "1506410000", name);  
 Ob0.*addAt*(ob, "1506410282", desc);  
 Ob0.*addAt*(ob, "1506923461", region);  
 Ob0.*addAt*(ob, "1506223120", cost);  
 Ob0.*addAt*(ob, "1506910189", partner);  
 Ob0.*addAt*(ob, "1506310181", type);  
 Ob0.*addOb*(mains, ob);  
 }  
 response.sendRedirect("/eldar?task=7");  
 }  
  
 public void lection3task8() throws Exception {  
 String descId = request.getParameter("did");  
 descId = descId == null ? "" : descId;  
 if(!descId.equals("")) {  
 Ob0.*delOb*(mains, descId, myId);  
 }  
 Map<String, String> regions = new HashMap<>();  
 Map<String, String> partners = new HashMap<>();  
 regionsAndPartners(regions, partners);  
 StringBuilder text = new StringBuilder();  
 text.append("<form method=\"get\" action=\"#\">" +  
 "ID <input type=\"text\" name=\"did\">" +  
 "<input type=\"submit\" value=\"Удалить\">" +  
 "<input type=\"hidden\" name=\"task\" value=\"8\"></form>");  
 printDescs(text, regions, partners);  
 }  
  
 public void regionsAndPartners(Map<String, String> regions,  
 Map<String, String> partners) throws Exception {  
 Obb filter = Ob0.*createFilter*(5);  
 Ob0.*addCondition*(filter, 1000004, Ob0.ComparisonType.*EQ*, "100410000050");  
 Ob0.*addCondition*(filter, 1005101368, Ob0.ComparisonType.*NEQ*, "");  
 Obb[] cities = Ob0.*getSrcObs*(mains, filter, 0, 0);  
 Obb[] prtns = Ob0.*getSrcObs*(mains, Ob0.*createFilter*(158), 0, 0);  
 regions.put("", "-");  
 partners.put("", "-");  
 for(Obb city : cities) {  
 regions.put(city.id, city.getAt("1000098"));  
 }  
 for(Obb partner : prtns) {  
 partners.put(partner.id, partner.getAt("1001211"));  
 }  
 }  
  
 public void printDescs(StringBuilder text, Map<String, String> regions,  
 Map<String, String> partners) throws Exception {  
 Obb[] descs = Ob0.*getSrcObs*(mains, Ob0.*createFilter*(506), 0, 0);  
 text.append("<table class=\"data\"><tr><td>№</td><td>ID</td><td>Название</td>" +  
 "<td>Описание</td><td>Регион</td>" +  
 "<td>Доп. оплаты</td><td>Бронирование у парнера</td><td>Тип</td></tr>");  
 int i = 0;  
 for(Obb desc : descs) {  
 text.append(String.*format*("<tr><td>%d</td><td>%s</td><td>%s</td><td>%s</td>" +  
 "<td>%s</td><td>%s</td><td>%s</td><td>%s</td></tr>",  
 ++i, (desc.id\_user == myId ? "<b style=\"color:green;\">" + desc.id + "<b>" : desc.id),  
 desc.getAt("1506410000"),  
 desc.getAt("1506410282"),  
 regions.get(desc.getAt("1506923461")),  
 desc.getAt("1506223120"),  
 partners.get(desc.getAt("1506910189")),  
 typeMap.get(desc.getAt("1506310181"))));  
 }  
 text.append("</table>");  
 printHtml("Описания экскурсий", text.toString());  
 }  
  
 public void lection4task1() throws Exception {  
 long time = 100000000L;  
 int expire = 180;  
 String[] types = new String[]{"С", "БНС", "НС"};  
 String costId = Util.*s2s*(request.getParameter("id"));  
 String foundOrCreated = null;  
 Obb ob = null;  
 if(!costId.equals("")) {  
 ob = Ob0.*fromBytes*(Ob3.*get*(mains, redis, redis, (prefix + costId).getBytes()));  
 if(ob == null) {  
 ob = Ob0.*getOb*(mains, costId);  
 if(ob == null) {  
 foundOrCreated = "<div style=\"text-align:center;color:red;\">" +  
 "Искомый объект не существует в базе PostgreSQL</div>";  
 } else {  
 Ob3.*puts*(mains, redis, (prefix + costId).getBytes(), Ob0.*toBytes*(ob),  
 time, false, expire, "");  
 foundOrCreated = "<div style=\"text-align:center;color:blue;\">" +  
 "Объект изъят из базы PostgreSQL и записан в REDIS</div>";  
 }  
 } else {  
 foundOrCreated = "<div style=\"text-align:center;color:green;\">Объект найден</div>";  
 }  
 }  
 String text = String.*format*("<form method=\"get\" action=\"#\">" +  
 "ID <input type=\"text\" name=\"id\" value=\"%s\"> " +  
 "<input type=\"submit\" value=\"Найти\">" +  
 "<input type=\"hidden\" name=\"task\" value=\"9\">" +  
 "</form>%s%s", costId, foundOrCreated == null ? "" : foundOrCreated, ob == null ? "" :  
 String.*format*("<table class=\"data\">" +  
 "<tr><td>Название</td><td>ID номера</td><td>Тип стоимости</td></tr>" +  
 "<tr><td>%s</td><td>%s</td><td>%s</td></tr>" +  
 "</table>",  
 ob.getAt(1000348),  
 ob.getAt(1000350),  
 types[Integer.*parseInt*(ob.getAt(1046222729))]));  
 printHtml("Redis - Один", text);  
 }  
  
 public void lection4task2() throws Exception {  
 long time = 100000000L;  
 int expire = 1;  
 String action = request.getParameter("action");  
 action = action == null || action.equals("") ? "find" : action;  
 String key = Util.*s2s*(request.getParameter("key"));  
 String value = null;  
 if(action.equals("find") && !key.equals("")) {  
 byte[] bytes = Ob3.*get*(mains, redis, redis, (prefix + key).getBytes());  
 value = bytes == null ? "" : new String(bytes, StandardCharsets.*UTF\_8*);  
 } else if (action.equals("create")) {  
 value = request.getParameter("value");  
 Ob3.*puts*(mains, redis, (prefix + key).getBytes(), value.getBytes());  
 } else if (action.equals("delete")) {  
 *//Ob3.del(mains, redis, prefix + key, redis);*  
byte[] bytes = Ob3.*get*(mains, redis, redis, (prefix + key).getBytes());  
 value = bytes == null ? "" : new String(bytes, StandardCharsets.*UTF\_8*);  
 Ob3.*puts*(mains, redis, (prefix + key).getBytes(), "".getBytes(),  
 time, false, expire, "");  
 }  
 String text = String.*format*("<form method=\"get\" action=\"#\"><table class=\"form\">" +  
 "<tr><td>Ключ</td><td><input type=\"text\" name=\"key\"></td></tr>" +  
 "<tr><td>Значение</td><td><input type=\"text\" name=\"value\"></td></tr>" +  
 "</table>" +  
 "<input type=\"submit\" value=\"Создать\">" +  
 "<input type=\"hidden\" name=\"task\" value=\"10\">" +  
 "<input type=\"hidden\" name=\"action\" value=\"create\">" +  
 "</form>%s" +  
 "<form method=\"get\" action=\"#\">" +  
 "Ключ <input type=\"text\" name=\"key\"> " +  
 "<input type=\"submit\" value=\"Найти\">" +  
 "<input type=\"hidden\" name=\"task\" value=\"10\">" +  
 "<input type=\"hidden\" name=\"action\" value=\"find\">" +  
 "</form>%s" +  
 "<form method=\"get\" action=\"#\">" +  
 "Ключ <input type=\"text\" name=\"key\"> " +  
 "<input type=\"submit\" value=\"Удалить\">" +  
 "<input type=\"hidden\" name=\"task\" value=\"10\">" +  
 "<input type=\"hidden\" name=\"action\" value=\"delete\">" +  
 "</form>%s",  
 action.equals("create") ? "<p style=\"text-align:center;color:green\">[" +  
 key + " : " + value + "] создано</p>" : "<br>",  
 !key.equals("") && action.equals("find") ?  
 !value.equals("")  
 ? "<p style=\"text-align:center;color:green\">["  
 + key + " : " + value + "]</p>"  
 : "<p style=\"text-align:center;color:red\">Объект с ключом "  
 + key + " не найден.</p>"  
 : "<br>",  
 action.equals("delete")  
 ? !value.equals("")  
 ? "<p style=\"text-align:center;color:green\">["  
 + key + "] удален</p>"  
 : "<p style=\"text-align:center;color:red\">Объект с ключом "  
 + key + " не найден.</p>"  
 : "<br>");  
 printHtml("Redis - Два", text);  
 }  
  
 public void lection4task3() throws Exception {  
 Obb[] obs = Ob0.*getSrcObs*(mains, Ob0.*createFilter*(36), 0, 0);  
 List<String> result = new ArrayList<>(obs.length);  
 String action = Util.*s2s*(request.getParameter("action"));  
 action = action.equals("") ? "postgre" : action;  
 long time = -1;  
 if(action.equals("postgre")) {  
 time = catchTime(() -> {  
 for(Obb ob : obs) {  
 result.add(Ob0.*getOb*(mains, ob.id).id);  
 }  
 });  
 } else if(action.equals("redis")) {  
 time = catchTime(() -> {  
 for(Obb ob : obs) {  
 Obb temp = Ob0.*fromBytes*(Ob3.*get*(mains, redis, redis, (prefix + ob.id).getBytes()));  
 if(temp != null) {  
 result.add(temp.id);  
 } else {  
 result.clear();  
 break;  
 }  
 }  
 });  
 } else if(action.equals("in\_redis")) {  
 long time0 = 1000000000L;  
 int expire = 3600;  
 for(Obb ob : obs) {  
 Ob3.*puts*(mains, redis, (prefix + ob.id).getBytes(), Ob0.*toBytes*(ob),  
 time0, false, expire, "");  
 }  
 } else {  
 throw new Exception("Плохой аргумент");  
 }  
 StringBuilder text = new StringBuilder();  
 text.append("<div style=\"text-align:center;\"><p>" +  
 "<a href=\"?task=11&action=postgre\">Выбрать из PostgreSQL</a> | " +  
 "<a href=\"?task=11&action=redis\">Выбрать из Redis</a> | " +  
 "<a href=\"?task=11&action=in\_redis\">Заполнить Redis</a></p>");  
 text.append(time == -1  
 ? String.*format*("<p style=\"color:green\">%d объектов добавлено в Redis.</p>",  
 obs.length)  
 : !result.isEmpty()  
 ? String.*format*("<p>Время выборки из %s: %d мс</p>",  
 action.equals("postgre") ? "PostgreSQL" : "Redis", time)  
 : "<h1 style=\"text-align:center;\">" +  
 "<a href=\"?task=11&action=in\_redis\">Заполните</a> Redis</h1>");  
 text.append("</div>");  
 if(!result.isEmpty()) {  
 text.append(String.*join*(", ", result));  
 }  
 printHtml("Redis - Три", text.toString());  
 }  
  
 public long catchTime(Procedure procedure) throws Exception {  
 long time = System.*currentTimeMillis*();  
 procedure.run();  
 return System.*currentTimeMillis*() - time;  
 }  
  
 public void lection4task4() throws Exception {  
 printHtml("Redis - Четыре",  
 "<div style=\"border:1px solid #ccc; width:900px; margin:0px auto; padding:15px;\">" +  
 "Целесообразно применять Redis в: <ul>" +  
 "<li>онлайн-магазинах для корзины;" +  
 "<li>онлайн-играх по типу шахмат для хранения ходов и состояния шахматной доски;" +  
 "<li>программах с видеоконференциями для буферизации видеопотоков;" +  
 "<li>в общем, применима в тех случаях, когда требуется быстродействие и безопасна возможная" +  
 " потеря данных в связи с отключением электричества на сервере." +  
 "</ul></div>");  
 }  
  
 public void lection5task1() throws Exception {  
 String userEMail = getUserEMail();  
 String tourId = request.getParameter("tour\_id");  
 String quoteId = request.getParameter("quote\_id");  
 StringBuilder text = new StringBuilder();  
 new StackPager()  
 .add(() -> true,  
 () -> "task=13",  
 () -> "Экскурсии",  
 (String link) -> toursHtml(text, userEMail))  
 .add(() -> NumberUtils.*isNumber*(tourId),  
 () -> "tour\_id=" + tourId,  
 () -> "Квоты экскурсии ID " + tourId,  
 (String link) -> quotesHtml(text, tourId))  
 .add(() -> NumberUtils.*isNumber*(quoteId),  
 () -> "quote\_id=" + quoteId,  
 () -> "Квота ID " + quoteId,  
 (String link) -> quoteEditHtml(text, quoteId))  
 .run(text);  
 printHtmlWithAuth(userEMail, "Авторизация - Один", text.toString());  
 }  
  
 public void toursHtml(StringBuilder text, String userEMail) throws Exception {  
 Obb tourFilter = Ob0.*createFilter*(36);  
 Obb quoteFilter = Ob0.*createFilter*(799);  
 Obb[] tours = Ob0.*getSrcObs*(mains, tourFilter, 0, 0);  
 Obb[] quotes = Ob0.*getSrcObs*(mains, quoteFilter, 0, 0);  
 Set<String> quotedTours = new HashSet<>();  
 for(Obb quote : quotes) {  
 quotedTours.add(quote.getAt(1799910177));  
 }  
 text.append("<table class=\"data\"><tr><td>№</td><td>Код экскурсии</td><td>Адрес</td></tr>");  
 int i = 0;  
 for(Obb tour : tours) {  
 text.append(String.*format*("<tr><td>%d</td><td>%s</td><td>%s</td></tr>",  
 ++i,  
 quotedTours.contains(tour.id) && userEMail != null  
 ? "<a href=\"?task=13&tour\_id=" + tour.id + "\">" + tour.id + "</a>"  
 : tour.id,  
 tour.getAt("1036423021") + "<br>" + tour.getAt("1036410028")));  
 }  
 text.append("</table>");  
 }  
  
 public void quotesHtml(StringBuilder text, String tourId) throws Exception {  
 Obb filter = Ob0.*createFilter*(799);  
 Ob0.*addCondition*(filter, 1799910177, Ob0.ComparisonType.*EQ*, tourId);  
 Obb[] quotes = Ob0.*getSrcObs*(mains, filter, 0, 0);  
 text.append("<table class=\"data\"><tr><td>№</td><td>ID</td><td>Дата</td><td>Количество</td></tr>");  
 int i = 0;  
 for(Obb quote : quotes) {  
 text.append(String.*format*("<tr><td>%d</td><td>%s</td><td>%s</td><td>%s</td></tr>", ++i,  
 String.*format*("<a href=\"?task=13&tour\_id=%s&quote\_id=%s\">%s</a>", tourId, quote.id, quote.id),  
 quote.getAt("1799510027"), quote.getAt("1799210047")));  
 }  
 text.append("</table>");  
 }  
  
 public void quoteEditHtml(StringBuilder text, String quoteId) throws Exception {  
 Obb quote = Ob0.*getOb*(mains, quoteId);  
 text.append(String.*format*("<form method=\"post\" action=\"#\">" +  
 "<table class=\"form\">" +  
 "<tr><td>Дата</td><td><input type=\"text\" name=\"date\" value=\"%s\"></td></tr>" +  
 "<tr><td>Количество</td><td><input type=\"text\" name=\"count\" value=\"%s\"></td></tr>" +  
 "</table><input type=\"hidden\" name=\"task\" value=\"13\">" +  
 "<input type=\"hidden\" name=\"tour\_id\" value=\"%s\">" +  
 "<input type=\"hidden\" name=\"quote\_id\" value=\"%s\">" +  
 "<input type=\"submit\" value=\"Обновить квоту\"></form>",  
 quote.getAt("1799510027"), quote.getAt("1799210047"),  
 quote.getAt("1799910177"), quote.id));  
 }  
  
 public void lection5task1post() throws Exception {  
 String tourId = request.getParameter("tour\_id");  
 String quoteId = request.getParameter("quote\_id");  
 String date = request.getParameter("date");  
 String count = request.getParameter("count");  
 Obb quote = Ob0.*getOb*(mains, quoteId);  
 Ob0.*addAt*(quote, "1799510027", date);  
 Ob0.*addAt*(quote, "1799210047", count);  
 quote.id\_user = myId;  
 Ob0.*edtOb*(mains, quote);  
 response.sendRedirect("eldar?task=13&tour\_id=" + tourId + "&quote\_id=" + quoteId);  
 }  
  
 public void lection5task2() throws Exception {  
 String userEMail = getUserEMail();  
 String action = request.getParameter("action");  
 String quoteId = request.getParameter("id");  
 StringBuilder text = new StringBuilder();  
 if(quoteId != null) {  
 Obb quote = Ob0.*getOb*(mains, quoteId);  
 DateFormat format = new SimpleDateFormat("dd.MM.yyyy");  
 Date date = format.parse(quote.getAt(1074100143));  
 if(date.after(new Date())) {  
 if ("add".equals(action)) {  
 Ob0.*addAt*(quote, 1074200033, "1");  
 text.append("<div style=\"text-align:center;color:green;\">Стоп добавлен</div>");  
 } else if ("del".equals(action)) {  
 Ob0.*addAt*(quote, 1074200033, "0");  
 text.append("<div style=\"text-align:center;color:green;\">Стоп убран</div>");  
 }  
 Ob0.*edtOb*(mains, quote);  
 } else {  
 text.append("<div style=\"text-align:center;color:red;\">Стоп не добавлен</div>");  
 }  
 }  
 Obb filter = Ob0.*createFilter*(74);  
 Obb[] quotes = Ob0.*getSrcObs*(mains, filter, 0, 0);  
 text.append("<table class=\"data\">");  
 text.append(String.*format*("<tr><td>№</td><td>Отель</td><td>Страна</td>" +  
 "<td>Дата начала</td><td>Дата окончания</td><td>Стоп</td>%s</tr>",  
 userEMail == null ? "" : "<td>Действие</td>"));  
 int i = 0;  
 for(Obb quote : quotes) {  
 text.append(String.*format*("<tr><td>%d</td><td>%s</td><td>%s</td><td>%s</td><td>%s</td><td>%s</td>%s</tr>",  
 ++i, Ob0.*getZn*(mains, quote.getAt(1000538), 1000127, 4),  
 Ob0.*getZn*(mains, quote.getAt(1000802), 1000000, 4),  
 quote.getAt(1074100142),  
 quote.getAt(1074100143),  
 quote.getAt(1074200033),  
 userEMail == null ? "" : "<td>" + (quote.getAt(1074200033).equals("0")  
 ? "<a href=\"?task=14&action=add&id=" + quote.id + "\">Добавить</a>"  
 : "<a href=\"?task=14&action=del&id=" + quote.id + "\">Удалить</a>") + "</td>"));  
 }  
 text.append("</table>");  
 printHtmlWithAuth(userEMail, "Авторизация - Два", text.toString());  
 }  
  
 public void lection5task3() throws Exception {  
 String userEMail = getUserEMail();  
 StringBuilder text = new StringBuilder();  
 text.append(hasRole("1001900012") + "<br>");  
 text.append(hasTitle("1007410000") + "<br>");  
 printHtmlWithAuth(userEMail, "Авторизация - Три", text.toString());  
 }  
  
 public void lection5task4() throws Exception {  
 String userEMail = getUserEMail();  
 String text;  
 if(userEMail == null) {  
 text = "<div style=\"text-align:center;\">Вы не авторизированы.</div>";  
 } else {  
 User whoami = AuthServlet.isAuth(request, response, mains, "meta");  
 text = String.format("<table class=\"form\">" +  
 "<tr><td>ID: </td><td>%s</td></tr>" +  
 "<tr><td>Тип ID: </td><td>%d</td></tr>" +  
 "<tr><td>Статус: </td><td>%s</td></tr>" +  
 "<tr><td>Логин: </td><td>%s</td></tr>" +  
 "<tr><td>Почта: </td><td>%s</td></tr>" +  
 "<tr><td>IP: </td><td>%s</td></tr>" +  
 "<tr><td>User-Agent: </td><td>%s</td></tr></table>",  
 whoami.id, Ob0.id2type(whoami.id), whoami.status, whoami.login, whoami.mail,  
 request.getHeader("X-Real-IP"), request.getHeader("User-Agent"));  
 }  
 printHtmlWithAuth(userEMail, "Авторизация - Четыре", text);  
 }  
  
 public void lection5task5() throws Exception {  
 String userEMail = getUserEMail();  
 String countryId = request.getParameter("country\_id");  
 String regionId = request.getParameter("region\_id");  
 String cityId = request.getParameter("city\_id");  
 String hotelId = request.getParameter("hotel\_id");  
 String roomId = request.getParameter("room\_id");  
 String nsId = request.getParameter("ns\_id");  
 String quoteId = request.getParameter("quote\_id");  
 StringBuilder text = new StringBuilder();  
 new StackPager()  
 .add(() -> true,  
 () -> "task=17",  
 () -> "Страны",  
 (String link) -> countriesHtml(text))  
 .add(() -> NumberUtils.isNumber(countryId),  
 () -> "country\_id=" + countryId,  
 () -> Ob0.getOb(mains, countryId).getAt(1000000) + ": регионы",  
 (String link) -> regionsHtml(text, countryId))  
 .add(() -> NumberUtils.isNumber(regionId),  
 () -> "region\_id=" + regionId,  
 () -> Ob0.getOb(mains, regionId).getAt(1000098) + ": города",  
 (String link) -> citiesHtml(text, link, regionId))  
 .add(() -> NumberUtils.isNumber(cityId),  
 () -> "city\_id=" + cityId,  
 () -> Ob0.getOb(mains, cityId).getAt(1000098) + ": отели",  
 (String link) -> hotelsHtml(text, link, cityId))  
 .add(() -> NumberUtils.isNumber(hotelId),  
 () -> "hotel\_id=" + hotelId,  
 () -> "Отель \"" + Ob0.getOb(mains, hotelId).getAt(1990410000) + "\"",  
 (String link) -> roomsHtml(text, link, hotelId))  
 .add(() -> NumberUtils.isNumber(roomId),  
 () -> "room\_id=" + roomId,  
 () -> "Номер \"" + Ob0.getOb(mains, roomId).getAt(1000168) + "\"",  
 (String link) -> nssHtml(text, link, roomId))  
 .add(() -> NumberUtils.isNumber(nsId),  
 () -> "ns\_id=" + nsId,  
 () -> "НС \"" + Ob0.getOb(mains, nsId).getAt(1000348) + "\"",  
 (String link) -> quotesHtmlV2(text, link, nsId))  
 .add(() -> NumberUtils.isNumber(quoteId),  
 () -> "quote\_id=" + quoteId,  
 () -> "Квота ID " + quoteId,  
 (String link) -> quoteEditHtmlV2(text, link, quoteId))  
 .run(text);  
 printHtmlWithAuth(userEMail, "Авторизация - Пять", text.toString());  
 }  
  
 public void countriesHtml(StringBuilder text) throws Exception {  
 Obb filter = Ob0.createFilter(4);  
 Obb[] countries = Ob0.getSrcObs(mains, filter, 0, 0);  
 Arrays.sort(countries, Comparator.comparing((Obb country) -> country.getAt(1000000)));  
 text.append("<table class=\"data\"><tr><td>№</td><td>Название</td></tr>");  
 int i = 0;  
 for(Obb country : countries) {  
 text.append(String.format("<tr><td>%d</td><td><a href=\"?task=17&country\_id=%s\">%s</a></td></tr>",  
 ++i, country.id, country.getAt(1000000)));  
 }  
 text.append("</table>");  
 }  
  
 public void regionsHtml(StringBuilder text, String countryId) throws Exception {  
 Map<String, String> regions = getRegions(countryId);  
 List<String> regionIds = intern.Utils.getKeysSortedByValue(regions, false);  
 text.append("<table class=\"data\"><tr><td>№</td><td>Название</td></tr>");  
 int i = 0;  
 for(String id : regionIds) {  
 text.append(String.format("<tr><td>%d</td><td>" +  
 "<a href=\"?task=17&country\_id=%s&region\_id=%s\">%s</a></td></tr>",  
 ++i, countryId, id, regions.get(id)));  
 }  
 text.append("</table>");  
 }  
  
 public void citiesHtml(StringBuilder text, String link, String regionId) throws Exception {  
 Obb filter = Ob0.createFilter(5);  
 Ob0.addCondition(filter, 1005101368, Ob0.ComparisonType.EQ, regionId);  
 Obb[] cities = Ob0.getSrcObs(mains, filter, 0, 0);  
 Arrays.sort(cities, Comparator.comparing((Obb city) -> city.getAt(1000098)));  
 text.append("<table class=\"data\"><tr><td>№</td><td>Название</td></tr>");  
 int i = 0;  
 for(Obb city : cities) {  
 text.append(String.format("<tr><td>%d</td><td><a href=\"%s&city\_id=%s\">%s</a></td></tr>",  
 ++i, link, city.id, city.getAt(1000098)));  
 }  
 text.append("</table>");  
 }  
  
 public void hotelsHtml(StringBuilder text, String link, String cityId) throws Exception {  
 Obb filter = Ob0.createFilter(990);  
 Ob0.addCondition(filter, 1990100059, Ob0.ComparisonType.EQ, cityId);  
 Obb[] hotels = Ob0.getSrcObs(mains, filter, 0, 0);  
 Arrays.sort(hotels, Comparator.comparing((Obb hotel) -> hotel.getAt(1990410000)));  
 text.append("<table class=\"data\"><tr><td>№</td><td>Название</td><td>НСы</td></tr>");  
 int i = 0;  
 for(Obb hotel : hotels) {  
 text.append(String.format("<tr><td>%d</td><td><a href=\"%s&hotel\_id=%s\">%s</a></td><td>%s</td></tr>",  
 ++i, link, hotel.id, hotel.getAt(1990410000),  
 String.join(", ", hotel.getAts(1990423125))));  
 }  
 text.append("</table>");  
 }  
  
 public void roomsHtml(StringBuilder text, String link, String hotelId) throws Exception {  
 Obb filter = Ob0.createFilter(21);  
 Ob0.addCondition(filter, 1000169, Ob0.ComparisonType.EQ, hotelId);  
 Obb[] rooms = Ob0.getSrcObs(mains, filter, 0, 0);  
 Arrays.sort(rooms, Comparator.comparing((Obb hotel) -> hotel.getAt(1000168)));  
 text.append("<table class=\"data\"><tr><td>№</td><td>Название</td></tr>");  
 int i = 0;  
 for(Obb room : rooms) {  
 text.append(String.format("<tr><td>%d</td><td><a href=\"%s&room\_id=%s\">%s</a></td></tr>",  
 ++i, link, room.id, room.getAt(1000168)));  
 }  
 text.append("</table>");  
 }  
  
 public void nssHtml(StringBuilder text, String link, String roomId) throws Exception {  
 Obb filter = Ob0.createFilter(46);  
 Ob0.addCondition(filter, 1000350, Ob0.ComparisonType.EQ, roomId);  
 Obb[] nss = Ob0.getSrcObs(mains, filter, 0, 0);  
 Arrays.sort(nss, Comparator.comparing((Obb ns) -> ns.getAt(1000348)));  
 text.append("<table class=\"data\"><tr><td>№</td><td>Название</td></tr>");  
 int i = 0;  
 for(Obb ns : nss) {  
 text.append(String.format("<tr><td>%d</td><td><a href=\"%s&ns\_id=%s\">%s</a></td></tr>",  
 ++i, link, ns.id, ns.getAt(1000348)));  
 }  
 text.append("</table>");  
 }  
  
 public void quotesHtmlV2(StringBuilder text, String link, String nsId) throws Exception {  
 Obb filter = Ob0.createFilter(990);  
 Ob0.addCondition(filter, 1000117, Ob0.ComparisonType.EQ, nsId);  
 Obb[] hotels = Ob0.getSrcObs(mains, filter, 0, 0);  
 Arrays.sort(hotels, Comparator.comparing((Obb hotel) -> hotel.getAt(1000127)));  
 text.append("<table class=\"data\"><tr><td>№</td><td>Название</td></tr>");  
 int i = 0;  
 for(Obb hotel : hotels) {  
 text.append(String.format("<tr><td>%d</td><td><a href=\"%s&hotel\_id=%s\">%s</a></td></tr>",  
 ++i, link, hotel.id, hotel.getAt(1000127)));  
 }  
 text.append("</table>");  
 }  
  
 public void quoteEditHtmlV2(StringBuilder text, String link, String quoteId) throws Exception {  
 Obb filter = Ob0.createFilter(990);  
 Ob0.addCondition(filter, 1000117, Ob0.ComparisonType.EQ, quoteId);  
 Obb[] hotels = Ob0.getSrcObs(mains, filter, 0, 0);  
 Arrays.sort(hotels, Comparator.comparing((Obb hotel) -> hotel.getAt(1000127)));  
 text.append("<table class=\"data\"><tr><td>№</td><td>Название</td></tr>");  
 int i = 0;  
 for(Obb hotel : hotels) {  
 text.append(String.format("<tr><td>%d</td><td><a href=\"%s&hotel\_id=%s\">%s</a></td></tr>",  
 ++i, link, hotel.id, hotel.getAt(1000127)));  
 }  
 text.append("</table>");  
 }  
  
 public static class StackPager {  
 private final List<Supplier<Boolean>> predicates = new ArrayList<>();  
 private final List<Supplier<String>> links = new ArrayList<>();  
 private final List<Supplier<String>> names = new ArrayList<>();  
 private final List<Consumer<String>> consumers = new ArrayList<>();  
  
 StackPager add(Supplier<Boolean> predicate, Supplier<String> link,  
 Supplier<String> name, Consumer<String> consumer) {  
 predicates.add(predicate);  
 links.add(link);  
 names.add(name);  
 consumers.add(consumer);  
 return this;  
 }  
  
 void run(StringBuilder text) throws Exception {  
 StringBuilder link = new StringBuilder();  
 StringBuilder name = new StringBuilder();  
 int n = 0;  
 while(++n < predicates.size() && predicates.get(n).get());  
 for(int i = 0; i < n; ++i) {  
 link.append(i == 0 ? "eldar?" : "&").append(links.get(i).get());  
 name.append(i == 0 ? "" : " -> ").append(i < n - 1 ? String.format("<a href=\"%s\">", link) : "")  
 .append(names.get(i).get()).append(i < n - 1 ? "</a>" : "");  
 }  
 text.append("<div style=\"text-align:center; margin-bottom:20px;\">").append(name).append("</div>");  
 consumers.get(n - 1).accept(link.toString());  
 }  
 }  
  
 public interface Supplier<T> {  
 T get() throws Exception;  
 }  
  
 public interface Consumer<T> {  
 void accept(T arg) throws Exception;  
 }  
  
 public void printHtmlWithAuth(String userEMail, String title, String text) throws Exception {  
 String task = request.getParameter("task");  
 authTask(userEMail != null);  
 StringBuilder output = new StringBuilder();  
 output.append("<div style=\"text-align:center;margin-bottom:20px;\">");  
 if(userEMail == null) {  
 output.append("<a href=\"?task=" + task + "&auth=need\">Авторизоваться</a>");  
 } else {  
 output.append("Привет, " + userEMail + "!");  
 }  
 output.append("</div>");  
 printHtml(title, output + text);  
 }  
  
 public void authTask(boolean isAuth) throws Exception {  
 String authParameter = request.getParameter("auth");  
 boolean doYouNeedAuth = authParameter != null && authParameter.equals("need");  
 boolean doYouNeedQuit = authParameter != null && authParameter.equals("quit");  
 if(!isAuth && doYouNeedAuth) {  
 AuthServlet.isAuth(request, response, mains, "meta");  
 }  
 if(isAuth && doYouNeedQuit) {  
 String task = request.getParameter("task");  
 response.setContentType("text/html; charset=UTF-8");  
 Cookie[] cookies = request.getCookies();  
 if(cookies != null) {  
 for (var cookie : cookies) {  
 cookie.setValue("");  
 cookie.setPath("/");  
 cookie.setMaxAge(0);  
 response.addCookie(cookie);  
 }  
 }  
 response.sendRedirect("eldar?task=" + task);  
 }  
 }  
  
 public boolean hasRole(String role) {  
 if(getUserEMail() == null) {  
 return false;  
 }  
 Obb obUser = getUser();  
 return Base.userHasRole(mains, Ob0.metaconnname, obUser, role);  
 }  
  
 public int hasTitle(String role) {  
 if(getUserEMail() == null) {  
 return -5;  
 }  
 User whoami = AuthServlet.isAuth(request, response, mains, "meta");  
 Obb obUser = PersonalPageServlet.getUserByComm(mains, whoami);  
 return Base.userHasRole(mains, Ob0.metaconnname, whoami, obUser, "100718258857",  
 request.getHeader("X-Real-IP"), request.getHeader("User-Agent"));  
 }  
  
 public Obb getUser() {  
 User whoami = AuthServlet.isAuth(request, response, mains, "meta");  
 return PersonalPageServlet.getUserByComm(mains, whoami);  
 }  
  
 public String getUserEMail() {  
 Cookie[] cookies = request.getCookies();  
 boolean A100 = false, Z100 = false;  
 String L = null;  
 if(cookies != null) {  
 for (var cookie : cookies) {  
 if (cookie.getName().equals("A100") && !cookie.getValue().equals("")) {  
 A100 = true;  
 }  
 if (cookie.getName().equals("Z100") && !cookie.getValue().equals("")) {  
 Z100 = true;  
 }  
 if (cookie.getName().equals("L") && !cookie.getValue().equals("")) {  
 L = cookie.getValue();  
 }  
 }  
 }  
 return A100 && Z100 ? L : null;  
 }  
  
 public void cheatCode() throws Exception {  
 response.setContentType("application/json; charset=UTF-8");  
 String type = request.getParameter("type");  
 String s\_begin = request.getParameter("begin");  
 String s\_count = request.getParameter("count");  
 String s\_id = request.getParameter("id");  
 String s\_atid = request.getParameter("at");  
 String s\_val = request.getParameter("val");  
 int begin, count;  
 Obb[] obbs;  
 if(NumberUtils.isNumber(s\_id)) {  
 obbs = new Obb[]{Ob0.getOb(mains, s\_id)};  
 } else {  
 if(!NumberUtils.isNumber(type)) {  
 return;  
 }  
 begin = !NumberUtils.isNumber(s\_begin) ? 0 : Integer.parseInt(s\_begin);  
 count = !NumberUtils.isNumber(s\_count) ? 0 : Integer.parseInt(s\_count);  
 Obb filter = Ob0.createFilter(Integer.parseInt(type));  
 if(NumberUtils.isNumber(s\_atid) && NumberUtils.isNumber(s\_val)) {  
 Ob0.addCondition(filter, Integer.parseInt(s\_atid), Ob0.ComparisonType.EQ, s\_val);  
 }  
 obbs = Ob0.getSrcObs(mains, filter, begin, count);  
 }  
 out.print("[");  
 int i = 0;  
 for(Obb obb : obbs) {  
 out.printf("{\"id\":\"%s\",\"id\_user\":\"%s\",", obb.id, obb.id\_user);  
 int j = 0;  
 for(String key : obb.zn.keySet()) {  
 out.printf("\"%s\":\"%s\"%s", key, obb.getAt(key), ++j < obb.zn.size() ? "," : "");  
 }  
 out.printf("}%s", ++i < obbs.length ? "," : "");  
 }  
 out.print("]");  
 }  
  
 public void cheatFunction() throws Exception {  
 */\*Obb ob = Ob0.getOb(mains, "103610006184");*  
 *ob.zn.remove("1799210047");*  
 *ob.zn.remove("1799510027");*  
 *ob.id\_user = myId;*  
 *Ob0.edtOb(mains, ob);*  
 *out.print(ob.zn);\*/*  
}  
  
 public void printHtml(String title, String text) {  
 response.setContentType("text/html; charset=UTF-8");  
 String task = request.getParameter("task");  
 StringBuilder output = new StringBuilder();  
 output.append(String.format("<!doctype html><html><head><title>%s</title><style>" +  
 "table { margin: 0px auto; }" +  
 "table.form tr td:first-child { text-align:right; }" +  
 "table.form tr td:last-child { text-align:left; }" +  
 "table.data { border: solid 1px #ccc; border-spacing: 3px;" +  
 "border-collapse: collapse; margin-bottom:10px; }" +  
 "table.data tr:first-child { font-weight:bold; }" +  
 "table.data td { border: solid 1px #ccc; padding: 5px; }" +  
 "form { margin: 0px auto; margin-bottom:20px; text-align:center;" +  
 "padding:10px; }" +  
 "ul { list-style: none; }" +  
 "ul li:before { content: \"—\"; position: relative; left: -5px; }" +  
 "</style></head>" +  
 "<body><div style=\"text-align:center;margin-bottom:20px;\">" +  
 "<table class=\"form\">", title));  
 int tasks = 0;  
 for(String key : pagesMap.keySet()) {  
 output.append(String.format("<tr>" +  
 "<td style=\"font-weight:bold;color:#072d78;padding-right:15px;\">" +  
 "%s</td><td>", key));  
 List<String> numbers = pagesMap.get(key);  
 for(int i = 1, n = numbers.size(); i <= n; ++i) {  
 if(String.valueOf(tasks + i).equals(task)) {  
 output.append(String.format("<b style=\"color:grey\">%s</b>%s",  
 numbers.get(i - 1), i < n ? " | " : ""));  
 } else {  
 output.append(String.format("<a href=\"?task=%d\">%s</a>%s",  
 tasks + i, numbers.get(i - 1), i < n ? " | " : ""));  
 }  
 }  
 output.append("</td></tr>");  
 tasks += numbers.size();  
 }  
 output.append(String.format("</table></div>%s</body></html>", text));  
 out.print(output);  
 }  
}