# Programming Test for CP Global Financial

Position: Java Developer

**Test 1**

Write Java code to find a 9 letter string of characters that contains only letters from

Acdegilmnoprstuw  
such that the hash(the\_string) is

956446786872726

if hash is defined by the following pseudo-code:

Int64 hash (String s) {

Int64 h = 7

String letters = "acdegilmnoprstuw"

for(Int32 i = 0; i < s.length; i++) {

h = (h \* 37 + letters.indexOf(s[i]))

}

return h

}

For example, if we were trying to find the 7 letter string where hash(the\_string) was 680131659347, the answer would be

"leepadg".

**Ans:**

**public class** Test {  
  
 **public static** String hash(**long** num) {  
 String letters = **"acdegilmnoprstuw"**;  
 **int** remainder = 0;  
 **int**[] pos = **new int**[10];  
 **char**[] result = **new char**[10];  
 **for** (**int** i = 9; i > 0; i--) {  
 remainder = (**int**) (num % 37);  
 pos[i] = remainder;  
 num = (num - remainder) / 37;  
 result[i - 1] = letters.charAt(pos[i]);  
 }  
 **return new** String(result);  
 }  
 **public static void** main(String... args){System.***out***.println(*hash*(Long.*parseLong*(**"956446786872726"**)));  
 }  
}

**OUTPUT** is **trellises**

**Test 2**

Please rewrite the following Spring controller to reduce duplication. Hint: use abstract classes.

CommonController:

@Controller

public class CommonController extends AbstractController {

@Autowired

private TeacherService teacherService;

@Autowired

private StudentService studentService;

@Autowired

private ParentService parentService;

@RequestMapping(value = {"/common/searchTeacher"}, method = RequestMethod.GET)

public @ResponseBody Object searchTeacher(@RequestParam(value = "term") String term) throws Exception {

PagingDto<Teacher> pagingDto = new PagingDto<Teacher>();

pagingDto.setResultsPerPage(null);

pagingDto.setSearch(term);

List<Teacher> teachers = teacherService.retrieveList(pagingDto, false).getResults();

List<Map<String, String>> ret = new ArrayList<Map<String, String>>();

for (Teacher teacher : teachers) {

Map<String, String> res = new HashMap<String, String>();

res.put("label", teacher.getExperience());

res.put("value", teacher.getId());

ret.add(res);

}

return ret;

}

@RequestMapping(value = {"/common/searchStudent"}, method = RequestMethod.GET)

public @ResponseBody Object searchStudent(@RequestParam(value = "term") String term) throws Exception {

PagingDto<Student> pagingDto = new PagingDto<Student>();

pagingDto.setResultsPerPage(null);

pagingDto.setSearch(term);

List<Student> students = studentService.retrieveList(pagingDto, false).getResults();

List<Map<String, String>> ret = new ArrayList<Map<String, String>>();

for (Student student : students) {

Map<String, String> res = new HashMap<String, String>();

res.put("label", student.getInterests()());

res.put("value", student.getId());

ret.add(res);

}

return ret;

}

@RequestMapping(value = {"/common/searchParent"}, method = RequestMethod.GET)

public @ResponseBody Object searchParent(@RequestParam(value = "term") String term) throws Exception {

PagingDto<Parent> pagingDto = new PagingDto<Parent>();

pagingDto.setResultsPerPage(null);

pagingDto.setSearch(term);

List<Parent> parents = parentService.retrieveList(pagingDto, false).getResults();

List<Map<String, String>> ret = new ArrayList<Map<String, String>>();

for (Parent parent : parents) {

Map<String, String> res = new HashMap<String, String>();

res.put("label", parent.getEducationHistory());

res.put("value", parent.getId());

ret.add(res);

}

return ret;

}

}

**Refactored Code :**

public abstract class AbstractController {

@GET

@Path("/search")

public Object search(String term) {

PagingDto<Person> pagingDto = new PagingDto<Person>();

pagingDto.setResultsPerPage(null);

pagingDto.setSearch(term);

List<Person> parents = psersonService.retrieveList(pagingDto, false).getResults();

List<Map<String, String>> ret = new ArrayList<Map<String, String>>();

for (Parent parent : parents) {

Map<String, String> res = new HashMap<String, String>();

res.put("label", getLabel(u)); // to be implemented in our subclasses

res.put("value", parent.getId());

ret.add(res);

}

return ret;

}

protected abstract String getLabel(Person person);

}

@Path("teacher") // Path :: /common/search/teacher

public class TeacherController extends AbstractController {

protected String getLabel(Teacher teacher) {

return teacher.getExperience();

}

}

@Path("student") // Path :: /common/search/student

public class StudentController extends AbstractController {

protected String getLabel(Student student) {

return student.getInterests();

}

}

@Path("parent") // Path :: /common/search/parent

public class ParentController extends AbstractController {

protected String getLabel(Parent parent) {

return parent.getEducationHistory();

}

}

Please email us with your solution. Include any code you used to solve the problem as an attachment. In the body of the email please explain why you would be a good fit for this job. If you have a website, send us the URL!