Run-Time Type Information

Laboratory Exercise

1. Write base class “Job”, which contains:
   - Protected:
     - Salary
     - Hours per week
   - Public:
     - Virtual void method that prints the salary and the amount of hours per week
2. Write “Baker” class, that inherits the Job class in “public” mode
   - Private:
     - Name of bakery
     - Breads per day
   - Public:
     - Constructor should take the arguments from base class with addition of the new variables given
     - Overwrite printing method from base class and print out the info about bakery.

3. Write “Secretary” class, that inherits the Job class in “public” mode
   - Private:
     - Name of the company
   - Public:
     - Constructor should take the arguments from base class with addition of the new variables given
     - Overwrite printing method from base class and print out the info about secretary

4. Write “Policeman” class, that inherits the Job class in “public” mode
   - Private:
     - Donuts eaten per day
   - Public:
     - Constructor should take the arguments from base class with addition of the new variables given
     - Overwrite printing method from base class and print out the info about policeman

5. Write a global function getJob()
   a. Take no arguments
   b. The function should be able to count the number of its calls (**static**)
   c. The function should be able to create and object of type Baker, Secretary, Policeman
   d. The type of a created object should be determined according to the number of function calls. Ex. 1st call object of type Baker should be created. On 2nd call object of type Secretary, On 3rd Policeman, On 4th Baker, On 5th Secretary…
   e. Function should return a pointer of type Job* with an address of created object.
6. In main function:

   6.1

   a. Create pointer of type **Job***
   b. Create pointer of type **Baker***
   c. Create a loop proceeds with these steps (3 iterations):
      - Assign the value of **getJob**() function to **Job*** pointer
      - Assign the address of pointer **Job*** to the **Baker*** (**dynamic_cast**)
      - After conversion use **Baker*** pointer to call printing method ex. **jobInfo**()

   Take in mind that conversion might be not possible, these situations should be handled.

   6.2

   1. Use pointer job* (created in 6.1) and assign to it the returned value of function getJob().
   2. Use the **typeid()** operator to check the type of currently pointed object. If the object is of Baker type, then call the printing method. If not, print on the screen the name of the type.
   3. These steps should be repeated in a loop. (3 iterations).