01/27/12 APCS – Monta Vista High School Search, Numbers & Exceptions Name_____

1. In the Java programming language, exceptions are instances of a(n):

(A) class
(B) interface
(C) primitive data type
(D) named constant
(E) enumeration

2. Suppose an array **arr** contains 97 different random values arranged in ascending order, and a binary search algorithm is used to find a target value. How many elements of the array will be examined when the target equals **arr[17]?**

(A) 3
(B) 4
(C) 5
(D) 6
(E) 7

3. The class **CourseList** provides methods that allow you to represent and manipulate a list of high school courses, but you are not concerned with how these operations work or how the list is stored in memory. You only know how to initialize and use **CourseList** objects and have no direct access to the implementation of the **CourseList** class or its private data fields. This is an example of

(A) method overloading
(B) polymorphism
(C) inheritance
(D) overriding
(E) encapsulation

4. The number of comparisons in Selection Sort in an array of n elements is roughly proportional to

(A) n
(B) n log n
(C) $(\log n)^2$
(D) n^2
(E) n^3

5. Consider the following code segment:

```
File goodfile = new File ("data.txt");
PrintWriter out = new PrintWriter (goodfile);
```

If, at runtime, **PrintWriter** cannot create a file with the given name, or if some other error occurs while opening or creating the file, an exception of type _____ will be thrown.

(A) FileNotFoundException
(B) InputOutputException
(C) PrinterException
(D) ExceptionalException
(E) InterruptedException

6. Consider an array of 800 (sorted) integers. What is the maximum number of steps that it can take to find this element (be sure to answer with an integer value):

(a) if we apply a linear search?

(b) if we apply a binary search?

7. Given the following numbers in the given base, convert the number to the indicated base.

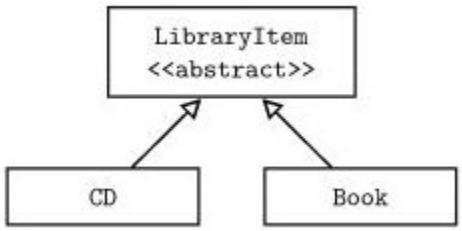
(a) Convert 749_{10} (a decimal number) to a hexadecimal number

(b) Convert 11101011010_2 (a binary number) to an octal number

(c) Convert $2C7_{16}$ (a hexadecimal number) to an octal number

(d) Convert 2691_{10} (a decimal number) to a binary number

8. Consider the following inheritance hierarchy diagram:



Suppose that, of the three classes shown in the diagram, only the **LibraryItem** class has a method called **getPrice** defined in it, whose specification is as follows:

//returns the price of this LibraryItem.
public double getPrice()

Now consider the following declaration that appears in a client program.

ArrayList <LibraryItem> libraryList = new ArrayList <LibraryItem> ();

Assume that **libraryList** is initialized with **LibraryItem** objects. Consider the following code segment that computes the total cost of all items stored in **libraryList**.

```
double total = 0.0;
for (LibraryItem item : libraryList)
        total += item.getPrice();
```

The use of **getPrice** in the above code segment is an example of:

(A) an overloaded method
(B) an overridden method
(C) polymorphism
(D) an inherited method
(E) an abstract method

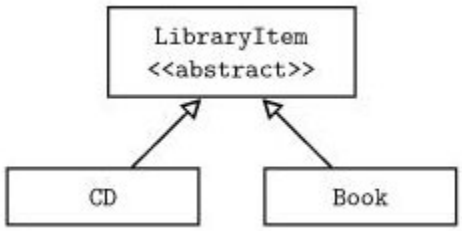
9. Why do you need to close an output text file before a program quits?

(A) To allow other programs to access the file
(B) To prevent other programs from accessing the file
(C) To make sure the data from a buffer is written to the file
(D) To prevent an IOException when the program exits
(E) To make sure that a breeze does not enter and blow all of your data away

10. When a Java program performs an illegal operation, a special event known as an

_____ occurs.

11. Consider the following inheritance hierarchy diagram:



Which of the following declarations will **not** cause an error? You may assume that each of the classes has a default constructor.

I. **LibraryItem item = new LibraryItem();**
II. **Book valedor = new LibraryItem();**
III. **LibraryItem mariah = new CD();**

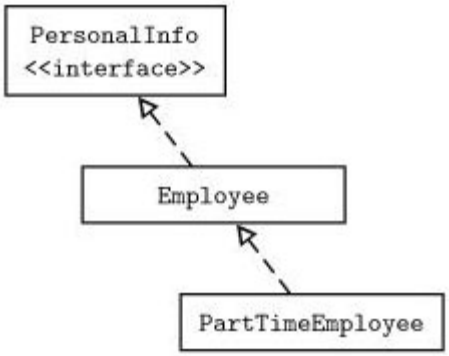
(A) I only
(B) II only
(C) III only
(D) II and III only
(E) I, II, and III

12. Consider the following code segment:

```
if (n != 0  &&  x/n > 100)
{
        statement1;
}
else
{
        statement2;
}
```

If **n** is of type **int** and has a value of **0** when the segment is executed, what will happen?

(A) An **ArithmeticException** will be thrown.
(B) A syntax error will occur.
(C) **statement1**, but not **statement2**, will be executed.
(D) **statement2**, but not **statement1**, will be executed.
(E) Neither **statement1** nor **statement2** will be executed.



13. Consider the class hierarchy diagram shown to the right.

PersonalInfo is an interface that is implemented by **Employee**. A **PartTimeEmployee** is-a **Employee**. Here is the declaration for **PersonalInfo**.

```
public interface PersonalInfo
{
        String getName();
        String getCity();
        boolean getCitizenStatus();
}
```

(a) Given the class hierarchy diagram shown above, write a complete class declaration for the class **Employee**, including implementation of methods and a constructor with parameters. An **Employee**, in addition to implementing **PersonalInfo**, has a data field to store annual salary, and an accessor method that returns the annual salary of the **Employee**. Write the **Employee** class below.

(b) Write a complete class declaration for the **PartTimeEmployee** class, given the class hierarchy shown above. A **PartTimeEmployee** has two additional data fields:

- A real number that indicates the fraction that the part-time employee works (for example, 0.5, 0.8, etc.)
- A boolean field that stores whether the employee is a union member or not.

There are three additional methods:

- An accessor that returns the real number fraction that the **PartTimeEmployee** works.
- An accessor that returns a value indicating whether the **PartTimeEmployee** is a union member or not.
- A mutator that switches the status of that employee's union membership, but only if the employee's salary is greater than $15,000. In other words, if his/her salary is greater than $15,000, then the union membership status should be "flipped" from union to nonunion or vice versa.

Write the **PartTimeEmployee** class below.

14. Consider a class, **NoteKeeper**, that is designed to store and manipulate a list of short notes. Here are some typical notes:

pick up drycleaning
special dog chow
dog registration
dentist Monday
study for APCS quiz

The incomplete class declaration is shown below:

```
public class NoteKeeper
{
    private ArrayList<String> noteList;

    // Postcondition:  Prints all notes in noteList, one per line.
    //                 Notes are numbered  1, 2, 3, …
    //                 Each number is followed by a period and a space.
    public void printNotes ( )
    {
        /*  to be implemented in part (a)  */
    }

    // Postcondition:  All notes with specified word have been removed from noteList, leaving the order of the
    //                 remaining notes unchanged.  If none of the notes in noteList contains word, the list remains
    //                 unchanged.
    public void removeNotes(String word)
    {
        /*  to be implemented in part (b)  */
    }

    // Constructor and other methods not shown . . .
}
```

(a) Write the **NoteKeeper** method **printNotes**. The **printNotes** method prints all of the notes in **noteList**, one per line, and numbers the notes, starting at 1. The output should look like this:

1. pick up drycleaning
2. special dog chow
3. dog registration
4. dentist Monday
5. study for APCS quiz

Complete method **printNotes** below.

```
// Postcondition:  Prints all notes in noteList, one per line.
//                 Notes are numbered  1, 2, 3, …
//                 Each number is followed by a period and a space.
public void printNotes ( )
```

(b) Write the **NoteKeeper** method **removeNotes**. Method **removeNotes** removes all notes from **noteList** that contain the word specified by the parameter. The ordering of the remaining notes should be left unchanged. For example, suppose that a **NoteKeeper** variable, **notes**, has a **noteList** containing

[pick up drycleaning, special dog chow, dog registration, dentist Monday, study for APCS quiz]

The method **notes.removeNotes("dog")** should modify the **noteList** of **notes** to be

[pick up drycleaning, dentist Monday, study for APCS quiz]

Complete method **removeNotes** below.

```
// Postcondition:  All notes with specified word have been removed from noteList, leaving the order of the
//                 remaining notes unchanged.  If none of the notes in noteList contains word, the list remains
//                 unchanged.
public void removeNotes(String word)
```