UIL COMPUTER SCIENCE
S-18 Practice Packet

Written by
Kirby Rankin

Edited by Nancy Barnard

Author Kirby Rankin brings over 25 years of teaching experience and has coached Computer Science for most of those years. His successes include three individual champions and six 2A team champions, and these were in a row from 2008 through 2013. He had many, many more competitors qualify for region and state over his years.

We are a small company that listens! If you have any questions or if there is an area that you would like fully explored, let us hear from you. We hope you enjoy this product and stay in contact with us throughout your academic journey.
~ President Hexco Inc., Linda Tarrant

HEXCO ACADEMIC
www.hexco.com
P.O. Box 199 • Hunt, Texas 78024
Phone: 800.391.2891 • Fax: 830.367.3824
Email: hexco@hexco.com

Standard Classes and Interfaces – Supplemental Reference

`class java.long.Object`

- `boolean equals )Object other)`
- `String toString ()`
- `Int hashCode ()`

`interface java.lang.Comparable<T>`

- `Int compareTo (T other)`
  Return value ,0 if this is less than other.
  Return value = 0 if this is equal to other.
  Return vale > 0 if this is greater than other.

`class java.lang.Integer implements Comparable<Integer>`

- `integer (int value)`
- `int intValue ()`
- `boolean equals (Object obj)`
- `String toString ()`
- `int compareTo (Integer anotherInteger)`
- `static in parseInt (String s)`
- `static int parseInt (String s, int radix)`

`class java.lang.Double implements Comparable<Double>`

- `Double (double value)`
- `double doubleValue ()`
- `boolean equals (Object obj)`
- `String toString ()`
- `Int compareTo (Double anotherDouble)`
- `static double parseDouble (String s)`

`class java.lang.String implements Comparable<String>`

- `int compareTo (String anotherString)`
- `boolean equals (Object obj)`
- `int length ()`
- `String substring (int begin, int end)`
  Returns the substring starting at index `begin` and ending at index `(end – 1)`.
- `String substring (int begin)`
  Returns `substring (from, length())`.
- `int indexOf (String str)`
  Returns the index within this string of the first occurrence of `str`. Returns `-1` if `str` is not found.
- `int indexOf (String str, int fromIndex)`
  Returns the index within this string of the first occurrence of `str`, starting the search at the specified index. Returns `-1` if `str` is not found.
- `charAt (int index)`
- `int indexOf (int ch)`
- `int indexOf (int ch, int fromIndex)`
- `String toLowerCase ()`
- `String toUpperCase ()`
- `String [] split (String regex)`
- `boolean matches (String regex)`

`class java.lang.Character`

- `static boolean isDigit (char ch)`
- `static boolean isLetter (char ch)`
- `static boolean isLetterOrDigit (char ch)`
- `static boolean isLowerCase (char ch)`
- `static boolean isUpperCase (char ch)`
- `static char toUpperCase (char ch)`
- `static char toLowerCase (char ch)`

`class java.lang.Math`

- `static int abs (int a)`
- `static double abs (double a)`
- `static double pow (double base, double exponent)`
- `static double sqrt (double a)`
- `static double ceil (double a)`
- `static double floor (double a)`
- `static double min (double a, double b)`
- `static double max (double a, double b)`
- `static int min (int a, int b)`
- `static int max (int a, int b)`
- `static long round (double a)`
- `static double random ()`
  Returns a `double` value with a positive sign, greater than or equal to `0.0` and less than `1.0`.

`Interface java.util.List<E>`

- `boolean add(E e)`
- `int size ()`
- `Iterator<E> listIterator()`
- `E get (int index)`
- `E set (int index, E e)`
  Replaces the element at `index` with the object `e`.
- `Void add (int index, E e)`
  Inserts the object `e` at position `index`, sliding elements at position `index` and higher to the right (adds 1 to their indices) and adjusts `size`.
- `E remove (int index)`
  Removes element from position `index`, sliding elements at position `(index + 1)` and higher to the left (subtracts `1` from their indices) and adjusts `size`.

`class java.util.ArrayList<E> implements List <E>`

`class java.util.LinkedList<E> implements List<E>, Queue<E>`

Methods in addition to the List methods:

- `Void addFirst (E e)`
- `Void addLast (E e)`
- `E getFirst ()`
- `E getLast ()`
- `E removeFirst ()`
- `E removeLast ()`

28. Which of the following must replace **<code>** in the Clock class to ensure that the instance variable `min` cannot be accessed from outside of the class?

    A. `private`
    B. `this`
    C. `public`
    D. `package`
    E. No additional code is necessary to restrict the access to `min`.

29. Which of the following should replace **<parameter>** in the `run` method?

    A. `minutes`
    B. `int min`
    C. `tMins`
    D. `int hour`
    E. `int minutes`

30. Assume that **<code>** and **<parameter>** have been filled in correctly. What is the output of the client code shown on the right?

```
Clock c1=new Clock(60);
Clock c2=new Clock(14,35);
Clock c3=new Clock();
c1.run(140);
c2.run(695);
c3.run(935);
out.println(c1+" "+c2+" "+c3);
```

    A. `1:40 7:35 9:35`
    B. `3:20 2:10 15:35`
    C. `3:20 2:10 3:35`
    D. `2:20 2:10 15:35`
    E. `2:20 7:35 3:35`

31. Which of the following is the best description of an insertion sort?

    A. Sorts a list by making repeated passes and with each pass swapping neighboring elements if the elements are not in sorted order.
    B. Recursively divides the list in half until each half contains just one element then merges each half in sorted order until the list is completely sorted.
    C. Sorts a list of values by repeatedly placing a new element into its proper location within a sorted sublist until the whole list is sorted.
    D. Selects an element to serve as the pivot value. Divides the list into two parts where all of the elements in one portion are smaller than the pivot value and all of the elements in the other portion are larger than the pivot value. Each portion is then recursively sorted using the same technique.
    E. Sorts a list by repeatedly finding the smallest element within the unsorted portion of the list and moving it to the end of the sorted portion of the list.

32. What is the worst case run time efficiency (Big O value) of a selection sort?

    A. O(1)
    B. O(n)
    C. O(log n)
    D. O(n log n)
    E. O(n^2)

7. What is the output of the code segment shown on the right?

```
long a=100;
byte b=12;
short c=82;
a*=c-b;
out.print(a);
```

A. `984`
B. `70`
C. `7000`
D. `9400`
E. Error. Type mismatch.

8. What is printed by the segment of code listed on the right?

```
String s1="integer";
String s2="double";
String s3="short";
if(s1.length()<s2.length())
     out.print(s1+" ");
else if(s1.length()>s2.length())
     out.print(s2+" ");
else
     out.print(s3+" ");
     out.print("byte");
```

A. `integer`
B. `double`
C. `double byte`
D. `integer byte`
E. `short`

9. Which of the following must replace **<code>** in the code segment shown on the right to ensure that the segment prints exactly five (5) asterisks (*)?

```
int x=10;
do {
     out.print("*");
     x-=2;
}while(<code>);
```

A. `x>=0`
B. `x<0`
C. `x>2`
D. `x>0`
E. `x<=1`

10. What is the output of the code segment shown on the right?

```
int[] z={5,0,3,1,4,2};
z[2]=z[1]+z[z.length-2];
out.print(z[2]);
```

A. `6`
B. `4`
C. `1`
D. `2`
E. `7`

22. Which of the following can replace **<code>** in the segment shown on the right to ensure that the code will compile and execute correctly?

```
Scanner scr=new Scanner("3.14
8.54 1.01 9.99 7.33");
double sum=0.0;
while(scr.hasNext())
   sum+=<code>;
out.print(sum);
```

A. `Double.parseDouble(scr.next())`
B. `scr.nextDouble()`
C. `scr.next()`
D. A and B
E. A, B, and C

23. Which of the following statements will return false given the declaration of `s` shown on the right?

```
String s="AAA123abc%@&";
```

A. `s.matches("A+\\d*\\w+\\W+")`
B. `s.matches("AAA123abc%@&")`
C. `s.matches("A*[123]*\\w{3}\\W+")`
D. `s.matches("A+\\d*\\w+[ABC]*\\W+")`
E. None of the above.

24. Which of the following expressions will result in a random whole number between 12 and 24 (inclusive) if `r` has been instantiated as a `Random` class object?

A. `12+r.nextInt(12)`
B. `12+r.nextInt(13)`
C. `r.nextInt()*12`
D. `r.nextInt()+12`
E. None of the above.

```
//Use the following code to answer questions 25 – 27.
public class NameAndNum implements Comparable<NameAndNum>{
      String name;
      int num;

      public NameAndNum(String name, int num) {
            this.name = name;
            this.num = num;
      }

      public String toString() {
            return name+" "+num;
      }

      public int compareTo(<code 1> o) {
            if(num<o.num)
                  return 1;
            else if(num>o.num)
                  return -1;
            else if(name.compareTo(<code 2>)<0)
                  return -1;
            else if(name.compareTo(<code 2>)>0)
                  return 1;
            else
                  return 0;
      }
}
```