QUEEN MARY, UNIVERSITY OF LONDON
DCS128 ALGORITHMS AND DATA STRUCTURES

Class Test Monday 27th March 2006 11.05-12.35

Please fill in your Examination Number here: .....

Student Number here: .....

All answers to this test should be written on the test sheet, but you may use spare paper for rough working. Answer as many of the questions as you can.

1) In the space below explain how the **selection sort** algorithm works (you do not need to give Java code for it, but your explanation should be a complete description). Also explain why the selection sort algorithm is categorised as $\mathbf{O(N^2)}$.

2) The Java code library contains two generic classes, **`ArrayList<E>`** and **`LinkedList<E>`**. They appear to work quite similarly. In the space below, explain how they are related, and how they differ, and say why Java provides these two different classes.

3) a) Explain briefly how a **static method** in Java differs from an **object method**.

b) Explain briefly the two uses of the keyword **`this`** in Java.

c) Explain what is meant by **dynamic binding** when a method is called in Java.

d) Explain what is meant by the **natural ordering** of a class of objects in Java.

4) Consider the following three Java classes which form part of a program for a fantasy world game:

```
class Being
{
 protected int strength;
 public Being(int s)
 {
  strength=s;
 }
 public int getStrength()
 {
  return strength;
 }
 public String toString()
 {
  return strength+" strength being";
 }
}


class Monster extends Being
{
 private String colour;
 private int legs;
 public Monster(int s,int l,String c)
 {
  super(s);
  legs=l;
  colour=c;
 }
 public String toString()
 {
  return colour+" "+legs+"-legged monster";
 }
}


class Hero extends Being
{
 private Weapon holds;
 private String name;
 public Hero(int s,String nm,Weapon w)
 {
  super(s);
  holds=w;
  name=nm;
 }
 public int getStrength()
 {
  return strength+holds.getPower();
 }
 public String toString()
 {
  return name+" holding "+holds;
 }
 public void pickup(Weapon w)
 {
  holds=w;
 }
}
```

Question continued on next page

Suppose we have variables `h` of type `Hero`, `m` of type `Monster`, `w` of type `Weapon`, `b` of type `Being`, `ab` of type `ArrayList<Being>`, `am` of type `ArrayList<Monster>`, `strs` of type `String[]` and `n` of type `int`.

For each of the following code fragments, say whether it is valid or invalid (where "invalid" code is code which would cause a Java compiler error):

Valid/Invalid

a)
```
h.pickup(w);
```
_____(a)

b)
```
b=h;
```
_____(b)

c)
```
if(h.getStrength()>m.getStrength())
   System.out.println(h);
```
_____(c)

d)
```
ab.remove(m);
```
_____(d)

e)
```
for(int i=0; i<am.size(); i++)
   System.out.println(am.get(i));
```
_____(e)

f)
```
m=b;
```
_____(f)

g)
```
for(int i=0; i<n; i++)
   am.add(new Monster("green"));
```
_____(g)

h)
```
for(int i=0; i<n; i++)
   ab.add(new Hero(10,strs[i],w));
```
_____(h)

i)
```
ab=am;
```
_____(i)

j)
```
am=ab;
```
_____(j)

k)
```
m=am.get(n);
```
_____(k)

l)
```
n=am.remove(m);
```
_____(l)

m)
```
m=am.remove(n);
```
_____(m)

n)
```
for(int i=0; i<ab.size(); i++)
   am.add(ab.get(i));
```
_____(n)

o)
```
for(int i=0; i<am.size(); i++)
   ab.add(am.get(i));
```
_____(o)

p)
```
for(int i=0; i<ab.size(); i++)
   strs[i]=ab.get(i).toString();
```
_____(p)

q)
```
if(am.get(n))
   System.out.println("Got monster "+n);
```
_____(q)

r)
```
if(ab.get(n) instanceof Hero)
   ab.get(n).pickup(w);
```
_____(r)

s)
```
h = (Hero) (ab.get(n));
```
_____(s)

t)
```
w = (Weapon) (strs[n]);
```
_____(t)

Note, this question will be marked on the basis of 1 mark for each correct answer, -1 mark for each incorrect answer.

5) This question uses the same code that was used for question 4).

a) Write a static method which takes an `ArrayList` of `Hero` or `Monster` objects, and returns a reference to whichever object in the `ArrayList` is the strongest (i.e. returns the highest value when `getStrength` is called on it).

b) Write a method `exchange` to go inside class `Hero` which works such that if `h1` and `h2` are two variables referring to objects of type `Hero`, calling `h1.exchange(h2)` will cause the objects to swap the values of their `holds` variables.

c) Write a method `equals` to go inside class `Monster` which overrides the default `equals` method, so that if `m1` and `m2` are two variables referring to objects of type `Monster`, `m1.equals(m2)` will return `true` if both objects have an equal `strength` variable, an equal `legs` variable and an equal `colour` variable, and `false` otherwise.