1(a) can only be assigned to once.

1(b) a per-class attribute; one copy of the attribute for the class

1(c) no, breaks encapsulation

1(d) x and y are the same object; x == y

1(e) many solutions are possible; one of the simplest is

```
@Override public int compareTo(Person other)     4 marks
{
  return this.age – other.age;                   2 marks
}
```

This solution is inconsistent because compareTo returns 0 if the ages are the same and the names are different (equals is false).

2(a) define state

2(b) define behavior

2(c) Class has poor encapsulation because of the public attribute.

1. A has no control over name
2. A cannot enforce any invariants on name
3. cannot change implementation of name without breaking client code

2(d)

1. private static attribute holding the singleton instance
2. private constructors
3. public static method to access the singleton

3(a)

100	main invocation
a	1
b	5

3(b)

500	swap invocation
a	1
b	5
tmp	1

500	main invocation
a	5
b	1
tmp	1

3(c) 1
5

4.

```
public final class StringUtil
{
  public static final String HELLO = "Hello, world.";       5 marks

  private StringUtil()
  { }

  // invalid: s null, n < 0                                    1 mark
  public static String repeat(String s, int n)               5 marks
  {                                                       body  4 marks
    String result = "";
    if(s != null && n > 0)
    {
      result = s;
      for(int i = 1; i < n; i++)
      {
        result = result + " " + s;
      }
    }
    return result;
  }
}
```

5.

add 2 static attributes 4 marks

```
private static String priorName = "Hammy"; // any reasonable name
private static int priorAge = 0;           // any reasonable age

public PetHamster()                                          2 marks
{
  this.name = PetHamster.priorName;                          1 mark
  this.ageInDays = PetHamster.priorAge;                      1 mark

}

public PetHamster(String name, int ageInDays)
{
  this.name = name;
  this.ageInDays = ageInDays;
  PetHamster.priorName = name;                               1 mark
  PetHamster.priorAge = ageInDays;                           1 mark
}
```

6(a)

```
public Point2d(int x, int y)                                    1 mark
{
  this.setX(x);                                                 2 marks
  this.setY(y);                                                 2 marks

}
```

6(b)

```
public Point2d()
{
  this(0, 0);                                                   2 marks
}
```

6(c)

```
public Point2d(Point2d other)                                   1 mark
{
  this(other.getX(), other.getY());                             2 marks
}
```

6(d)

```
@Override public boolean equals(Object obj)                     2 marks
{
  boolean eq = false;
  if(obj != null && this.getClass() == obj.getClass())          2 marks
  {
    Point2d other = (Point2d) obj;                              1 mark
    eq = this.getX() == other.getX() &&                         4 marks
         this.getY() == other.getY();
  }
  return eq;                                                    1 mark
}
```