Com S 228

Fall 2016

Exam 1 Sample Solution

1.

`BigCat c = new SiberianTiger("Sergei", 600);` `c.eat();`	compile error: The method eat() is undefined for the type Bigcat
`Tiger vijay = new BengalTiger("Vijay", 570);` `vijay.showStripes();`	Dense

`Tiger t;`
`t = new Tiger("Vijay", 570);`
`t = new Lion("Mufasa", 550);`
	compile error: Cannot instantiate the type Tiger
	
	compile error: Cannot convert from Lion to Tiger

`Sound s = new Lion("Simba", 500);` `((Lion) s).speak();`	Roar!

`BigCat c = new Lion("Nala", 400);`
`Lion twin = ((Lion) c).makeClone();`
`System.out.println(twin.getName());`
	Nala

`Sound s;`
`BengalTiger b = new BengalTiger("Vijay", 570);`
`s = b;`
`Lion t = new Lion("Mufasa", 550);`
`t = (Lion) ((BigCat) s);`
	ClassCastException

`Lion[] pride = new Lion[2];`
`pride[0] = new Lion("Simba", 500);`
`pride[1] = new Lion("Nala", 400);`
`LionPride serengeti = new LionPride`
`("Serengeti", pride);`
`System.out.println((serengeti.getLion`
`(serengeti.getSize()-2)).getName());`
	Simba

2a)

```
@Override
public boolean equals(Object o)
{
	if (o == null || o.getClass() != getClass())
	{
		return false;
	}

	// typecast o to Tiger so that we can compare data members
	Tiger t = (Tiger) o;

	// Compare the data members and return accordingly
	return t.weight == weight
			&& (t.name == name || t.name != null
			&&  t.name.equals(name));
}
```

b)

```
@Override
public LionPride clone()
{
	try
	{
		LionPride copy = (LionPride) super.clone();

		// Object.clone() copies fields, now make it into deep copy
		copy.lions = new Lion[size];
		for (int i = 0; i < size; ++i)
		{
			copy.lions[i] = lions[i].makeClone();
		}
		return copy;
	}
	catch (CloneNotSupportedException e)
	{
		// should never happen...
		return null;
	}
}
```

c)

```
@Override
public int compare(BigCat c1, BigCat c2)
{
    return c1.getWeight() - c2.getWeight();
}
```

3a)

i) $n - 1$ or $O(n)$
ii) $n - i - 1$ or $O(n)$
iii) $O(1)$
iv) $O(n^2)$

b)

i) $n/2$ or $O(n)$
ii) $O(n \log n)$
iii) $O(n^2 \log n)$
iv) $O(n^2 \log n)$

c)

i) $\log_4 n$
ii) $O(\log n)$

d) $O(n)$

4a) C: Merge Sort
b) B: Insertion Sort
c) A: Selection Sort
d) D: Quicksort
e) A: Selection Sort
f) B: Insertion Sort