SALMAN AHMED KHAN
19K-1043 BS SE (A)

Q#1

Answer: 1) Because we can not able to find which function will be execute as it can determined at only runtime.

2). If virtual function is present in the class then that class becomes abstract and if a developer forgets to override the virtual function in child class then the child class will also becomes abstract class.

3). We can not make objects of that class in which virtual function is created.

Q#2

Answer: Java treat method overridding as runtime polymorphism because it can not known at compile time that which function is called, whether parent or child class. It can only be determined

(2)

19K-1043

at runtime execution of program. which is based on the type of object which refer at the calling time of object.

Q#3

i). Overloading

```
clars Car{
        string make, color;
        int model;
       pulic:
        Car(){ }.
       void OverCarLoad(int m)
       {      model = m; }
       void OverCarLoad (string mk, string c)
       {      make = mk;
              color = c;
       }
};
```

ii). Overriding

```
clars Car {
    string make, color;
    int model;
public:
    Car () { }
    void seldata (string mk, string col,
                  int md )
    {
        make = mk;  color = col;
        model = md;
        cout << " Seldata in Car" ;
    }
};
clars Sport : public Car {
    public:
        Sport () { }
        void seldata (string mk, string col,
                      int md)
        { make = mk; color = col;
          if ( md < 500)
              { model = md; }
          else
              model = 100;  } .
};
```

iii). Virtual function

```
class animal {
  public:
      virtual void sound () = 0;
      virtual void food (void)
      {  cout << "Animal food"; }
};
```

```
class dog : public animal {
  public:
      void sound (void)
      {  cout << "Dog Bark Loud"; }
      void food (void)
      {  cout << "Dog eats Meat" ; }
};
```