```
// the following example displays the case where the derived
// class programmer can add new features
#include<iostream>
using namespace std;
// the following class has no properties
// hence if an object of the following class is created, zero byte
// memory cannot be allocated,
// hence for the sake of memory allocation, the allocation
// will be one byte
// Conclusion : Size of an object created from a propertyless
// class is of one byte. ( that byte is useless)
class Movie
{
public:
void start()
{
cout<<"Welcome"<<endl;
}
void interval()
{
cout<<"Interval - Have Coffee for Rs.50/-"<<endl;
}
void end()
{
cout<<"Thank you - come again"<<endl;
}
};
// the following class is being written by some other programmer
class JungleBook:public Movie
{
public:
void reelOne()
{
cout<<"Mowgli enters the jungle"<<endl;
}
void reelTwo()
{
cout<<"Bagheera saves Mowgli"<<endl;
}
};
// the following code is being written by some other programmer
int main()
{
JungleBook j; // 1 byte object of propertyless class
j.start();
j.reelOne();
j.interval();
j.reelTwo();
j.end();
return 0;
}
```

```
// the following example displays the case where the derived
// class programmer doesn't like a functionality of a base class
// and decides to override it
#include<iostream>
using namespace std;
class Movie
{
public:
void start()
{
cout<<"Welcome"<<endl;
}
void interval()
{
cout<<"Interval - Have Coffee for Rs.50/-"<<endl;
}
void end()
{
cout<<"Thank you - come again"<<endl;
}
};
// the following class is being written by some other programmer
// he wants that when interval gets called, a message related to
// Pepsi should get printed
class JungleBook:public Movie
{
public:
void reelOne()
{
cout<<"Mowgli enters the jungle"<<endl;
}
void reelTwo()
{
cout<<"Bagheera saves Mowgli"<<endl;
}
// to override, redefine the method of base class in derived
// class, keep name, signature and return type same
void interval()
{
cout<<"Interval - Have Pepsi for Rs.45/-"<<endl;
}
};
```

```
// the following code is being written by some other programmer
int main()
{
JungleBook j; // 1 byte object of propertyless class
j.start();
j.reelOne();
j.interval();
j.reelTwo();
j.end();
return 0;
}
```

```
// the following example displays the case where the derived
// class programmer doesn't like a functionality of a base class
// and decides to override it
#include<iostream>
using namespace std;
class Movie
{
public:
void start()
{
cout<<"Welcome"<<endl;
}
void interval()
{
cout<<"Interval - Have Coffee for Rs.50/-"<<endl;
}
void end()
{
cout<<"Thank you - come again"<<endl;
}
};
// the following class is being written by some other programmer
// he wants that when interval gets called, Whatever is being
// printed should get printed along with a message related to
// Pepsi
class JungleBook:public Movie
{
public:
void reelOne()
{
cout<<"Mowgli enters the jungle"<<endl;
}
void reelTwo()
{
cout<<"Bagheera saves Mowgli"<<endl;
}
void interval()
{
Movie::interval();
cout<<"and have Pepsi for Rs.45/-"<<endl;
}
};
// the following code is being written by some other programmer
int main()
{
JungleBook j; // 1 byte object of propertyless class
j.start();
j.reelOne();
j.interval();
j.reelTwo();
j.end();
return 0;
}
```

Topic of method overriding is completed

The following code won't compile

```
#include<iostream>
using namespace std;
class TV
{
private:
int price;
public:
void askinfo()
{
cout<<"Enter price of TV";
cin>>price;
}
void printinfo()
{
cout<<"Price of TV is "<<price<<endl;
}
};

class Radio
{
private:
int price;
public:
void askinfo()
{
cout<<"Enter price of Radio";
cin>>price;
}
void printinfo()
{
cout<<"Price of Radio is "<<price<<endl;
}
};
int main()
{
TV t;
t.askinfo();
Radio r;
r.askinfo();
int z=t.price+r.price;
cout<<"Total cost of TV and Radio is "<<z<<endl;
return 0;
}
```

The above code is not getting compiled, because main cannot access the price property of the TV and the Radio as it is private.

Try 1) What if we write a method in TV and compute the sum in it. Is it possible to do so ?
No, cannot be done, because the price property of Radio cannot be accessed in TV class
Try 2) What if we write a method in Radio and compute the sum in it. Is it possible to do so ?
No, cannot be done, because the price property of TV cannot be accessed in Radio class
Try 3) What if we inherit Radio from TV and then write a method in Radio with logic to compute the sum. Is it possible ?
No, because first thing TV and Radio are not related, so it is wrong to inherit Radio from TV as no relationship exists between them, and still if you inherit, it is not possible as the methods of derived class cannot access the private members of the base class.
Try 4)

```
#include<iostream>
using namespace std;
class TV
{
private:
int price;
public:
void askinfo()
{
cout<<"Enter price of TV";
cin>>price;
}
void printinfo()
{
cout<<"Price of TV is "<<price<<endl;
}
};

class Radio
{
private:
int price;
public:
void askinfo()
{
cout<<"Enter price of Radio";
cin>>price;
}
void printinfo()
{
cout<<"Price of Radio is "<<price<<endl;
}
};
void addPrice(TV &e,Radio &f)
{
int z;
z=e.price+f.price;
cout<<"Total cost of TV and Radio is "<<z<<endl;
}
int main()
{
TV t;
t.askinfo();
Radio r;
r.askinfo();
addPrice(t,r);
return 0;
}
```

The above code will not get compiled, because addPrice cannot access the private members of the TV and Radio class. So all 4 possible solutions failed

---

```
#include<iostream>
using namespace std;
class TV
{
private:
int price;
public:
void askinfo()
{
cout<<"Enter price of TV";
cin>>price;
}
void printinfo()
{
cout<<"Price of TV is "<<price<<endl;
}
friend void addPrice(TV &,Radio &);
};

class Radio
{
private:
int price;
public:
void askinfo()
{
cout<<"Enter price of Radio";
cin>>price;
}
void printinfo()
{
cout<<"Price of Radio is "<<price<<endl;
}
friend void addPrice(TV &,Radio &);
};
void addPrice(TV &e,Radio &f)
{
int z;
z=e.price+f.price;
cout<<"Total cost of TV and Radio is "<<z<<endl;
}
int main()
{
TV t;
t.askinfo();
Radio r;
r.askinfo();
addPrice(t,r);
return 0;
}
```

The above code will not get compiled, because when the compile starts checking the code from the first line to last line, and it will reach the friend void addPrice(TV &,Radio &) in TV class, till that point the compiler knows nothing about data type (Radio)
Solution: Declare Radio class as done in the following example

```
#include<iostream>
using namespace std;
class Radio;
class TV
{
```

```
private:
int price;
public:
void askinfo()
{
cout<<"Enter price of TV";
cin>>price;
}
void printinfo()
{
cout<<"Price of TV is "<<price<<endl;
}
friend void addPrice(TV &,Radio &);
};

class Radio
{
private:
int price;
public:
void askinfo()
{
cout<<"Enter price of Radio";
cin>>price;
}
void printinfo()
{
cout<<"Price of Radio is "<<price<<endl;
}
friend void addPrice(TV &,Radio &);
};
void addPrice(TV &e,Radio &f)
{
int z;
z=e.price+f.price;
cout<<"Total cost of TV and Radio is "<<z<<endl;
}
int main()
{
TV t;
t.askinfo();
Radio r;
r.askinfo();
addPrice(t,r);
return 0;
}
```

How many programmers are involved in the above example where 4 definitions have been created

1) TV class
2) Radio class
3) addPrice function
4) main function

Ans : Only 2 programmer, one is writing main and the rest is being written by another programmer.

When should the feature of friend function be used ?
When we want to perform operations on properties of non related entities(classes)
